

```

1: *****
2: Constructor Function and Procedure List of maXbox 3.9.9
3: *****
4:
5: //////////////////////////////////////
6: ref Help Extraxt of EXE Functions of maxbox3.exe BigBitBox API HEX in BOX
7: -----
8:
9: File Size of EXE:19942400 V3.9.9.94 April 2014 To EKON/BASTA/JAX/IBZ/SWS
10: *****Now the Funclist*****
11: Funclist Function : 12130 //10766 //10165 (7648)
12: *****Now the Proclist*****
13: Proclist Procedure Size is: 7566 //6792 //6401 4752
14: *****Now Constructors*****
15: Constructlist Constructor Size is: 1240 //995 //
16: def head:max: maXbox7: 29.03.2014 19:11:55
17: file E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt
18: doc file: maxbox_extract_funclist399_.txt (sort function list)
19: -----
20: Funclist total Size all is: 20936! Constructor, Function and Procedure
21: AExtraxt of EXE Functions of maxbox3.exe, locs of file = 19910
22: ASize of EXE: 19942400 (16586240) (13511680) (13023744)
23: SHA1 Hash of maXbox 3.9.9.94: 780F1F749EFEFB5C0FAA09C51B6AB048C6A00CC6
24:
25: -----
26: -----
27: //////////////////////////////////////
28:
29:
30: FUNCTION Metric of Script: 256_findfunctions2_of_EXE.txt
31: Function *****Now the Funclist*****
32: function GetResStringChecked(Ident: string; const Args: array of const): string
33: function ( Index : Longint) : Integer
34: function (Command: Word; Data: Longint; var CallHelp: Boolean): Boolean
35: function _CheckAutoResult( ResultCode : HRESULT ) : HRESULT
36: function _T(Name: tbtString): Variant;
37: function ABNFToText(const AText : String) : String
38: function Abs(e : Extended) : Extended;
39: function Ackermann( const A, B : Integer) : Integer
40: function AcquireLayoutLock : Boolean
41: function ActionByName( const AName : string) : TWebActionItem
42: function ACTIVEBUFFER : PCHAR
43: function Add : TAggregate
44: function Add : TCollectionItem
45: function Add : TColumn
46: function Add : TComboExItem
47: function Add : TCookie
48: function Add : TCoolBand
49: function Add : TFavoriteLinkItem
50: function Add : TFileTypeItem
51: function Add : THeaderSection
52: function Add : THTMLTableColumn
53: function Add : TIdEmailAddressItem
54: function Add : TIdMessagePart
55: function Add : TIdUserAccount
56: function Add : TListColumn
57: function Add : TListItem
58: function Add : TStatusPanel
59: function Add : TTaskDialogBaseButtonItem
60: function Add : TWebActionItem
61: function Add : TWorkArea
62: function Add( AClass : TClass) : Integer
63: function Add( AComponent : TComponent) : Integer
64: function Add( AItem, AData : Integer) : Integer
65: function Add( AItem, AData : Pointer) : Pointer
66: function Add( AItem, AData : TObject) : TObject
67: function Add( AObject : TObject) : Integer
68: function Add( const Access, Count : Cardinal; const Offset : Int64) : Integer
69: function Add( const S : WideString) : Integer
70: function Add( Image, Mask : TBitmap) : Integer
71: function Add( Index : Longint; const Text : string) : Longint
72: function Add( Sibling : TTreeNode; const S : string) : TTreeNode
73: function Add(const S: string): Integer
74: function Add(S: string): Integer;
75: function AddAt( const Access, Count : Cardinal; const Offset : Int64; const Address: Pointer) : Integer
76: function ADDCHILD : TFIELDDEF
77: function AddChild( Index : Longint; const Text : string) : Longint
78: function AddChild( Parent : TTreeNode; const S : string) : TTreeNode
79: function AddChildFirst( Parent : TTreeNode; const S : string) : TTreeNode
80: function AddChildObject( Index : Longint; const Text : string; const Data : Pointer) : Longint
81: function AddChildObject( Parent : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
82: function AddChildObjectFirst( Parent : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
83: function ADDFIELDDEF : TFIELDDEF
84: function AddFileExtIfNecessary( AFileName, AExt : string) : string
85: function AddFirst( Sibling : TTreeNode; const S : string) : TTreeNode
86: function AddIcon( Image : TIcon) : Integer
87: function AddImage( Value : TCustomImageList; Index : Integer) : Integer
88: function ADDINDEXDEF : TINDEXDEF
89: function AddItem(const Caption:String;const ImageIdx,SelectImageIdx,OverlayImageIdx,
Indent:Int;Data:Ptr):TComboExItem

```

```

90: Function AddItem( Item : THeaderSection; Index : Integer) : THeaderSection
91: Function AddItem( Item : TListItem; Index : Integer) : TListItem
92: Function AddItem( Item : TStatusPanel; Index : Integer) : TStatusPanel
93: Function AddMapping( const FieldName : string) : Boolean
94: Function AddMasked( Image : TBitmap; MaskColor : TColor) : Integer
95: Function AddModuleClass( AClass : TComponentClass) : TComponent
96: Function AddModuleName( const AClass : string) : TComponent
97: Function AddNode(Node,Relative: TTreeNode;const S : string;Ptr:Pointer;Method:TNodeAttachMode):TTreeNode
98: Function AddObject( const S : WideString; AObject : TObject) : Integer
99: Function AddObject( Index : LongInt; const Text : string; const Data : Pointer) : LongInt
100: Function AddObject( Sibling : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
101: function AddObject(S:String;AObject:TObject):integer
102: Function AddObjectFirst( Sibling : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
103: Function AddParameter : TParameter
104: Function AddParamsSQLForDetail(Params:TParams;SQL:WideStr;Native:Bool;QuoteChar:WideString):WideString
105: Function Addr64ToAddr32(const Value: TJclAddr64): TJclAddr32;
106: Function Addr32ToAddr64(const Value: TJclAddr32): TJclAddr64;
107: function AdjustLineBreaksS(const S: string): string
108: TTextLineBreakStyle', '(tlbsLF, tlbsCRLF)
109: Function AdjustLineBreaks(const S: string; Style: TTextLineBreakStyle): string;
110: Function AllData : string
111: function AllocMemCount: integer;
112: function AllocMemSize: integer;
113: Function AllocPatternBitmap( BkColor, FgColor : TColor) : TBitmap
114: Function AllowRegKeyForEveryone( Key : HKEY; Path : string) : Boolean
115: Function AlphaComponent( const Color32 : TColor32) : Integer
116: Function AlphaSort : Boolean
117: Function AlphaSort( ARecurse : Boolean) : Boolean
118: Function AnsiCat( const x, y : AnsiString) : AnsiString
119: Function AnsiCompareFileName( S1, S2 : string) : Integer
120: function AnsiCompareFileName(const S1: string; const S2: string): Integer)
121: Function AnsiCompareStr( S1, S2 : string) : Integer
122: function AnsiCompareStr(const S1: string; const S2: string): Integer;)
123: Function AnsiCompareText( S1, S2 : string) : Integer
124: function AnsiCompareText(const S1: string; const S2: string): Integer;)
125: Function AnsiContainsStr( const AText, ASubText : string) : Boolean
126: Function AnsiContainsText( const AText, ASubText : string) : Boolean
127: Function AnsiCopy( const src : AnsiString; index, count : Integer) : AnsiString
128: Function AnsiDequotedStr( S : string; AQuote : Char) : string
129: Function AnsiEndsStr( const ASubText, AText : string) : Boolean
130: Function AnsiEndsText( const ASubText, AText : string) : Boolean
131: Function AnsiExtractQuotedStr( var Src : PChar; Quote : Char) : string
132: function AnsiExtractQuotedStr(var Src: PChar; Quote: Char): string)
133: Function AnsiIndexStr( const AText : string; const AValues : array of string) : Integer
134: Function AnsiIndexText( const AText : string; const AValues : array of string) : Integer
135: Function AnsiLastChar( S : string) : PChar
136: function AnsiLastChar(const S: string): PChar)
137: Function AnsiLeftStr( const AText : AnsiString; const ACount : Integer) : AnsiString
138: Function AnsiLowerCase( S : string) : string
139: Function AnsiLowercase(s : String) : String;
140: Function AnsiLowerCaseFileName( S : string) : string
141: Function AnsiMatchStr( const AText : string; const AValues : array of string) : Boolean
142: Function AnsiMatchText( const AText : string; const AValues : array of string) : Boolean
143: Function AnsiMidStr( const AText : AnsiString; const AStart, ACount : Integer) : AnsiString
144: Function AnsiPos( const src, sub : AnsiString) : Integer
145: Function AnsiPos( Substr, S : string) : Integer
146: function AnsiPos(const Substr: string; const S: string): Integer;)
147: Function AnsiQuotedStr( S : string; Quote : Char) : string
148: Function AnsiReplaceStr( const AText, AFromText, AToText : string) : string
149: Function AnsiReplaceText( const AText, AFromText, AToText : string) : string
150: Function AnsiResemblesText( const AText, AOther : string) : Boolean
151: Function AnsiReverseString( const AText : AnsiString) : AnsiString
152: Function AnsiRightStr( const AText : AnsiString; const ACount : Integer) : AnsiString
153: function AnsiSameCaption(const Text1: string; const Text2: string): Boolean)
154: Function AnsiSameStr( S1, S2 : string) : Boolean
155: function AnsiSameStr(const S1: string; const S2: string): Boolean)
156: Function AnsiSameText( const S1, S2 : string) : Boolean
157: Function AnsiSameText( S1, S2 : string) : Boolean
158: function AnsiSameText(const S1: string; const S2: string): Boolean)
159: Function AnsiStartsStr( const ASubText, AText : string) : Boolean
160: Function AnsiStartsText( const ASubText, AText : string) : Boolean
161: Function AnsiStrComp( S1, S2 : PChar) : Integer
162: function AnsiStrComp(S1: PChar; S2: PChar): Integer)
163: Function AnsiStrIComp( S1, S2 : PChar) : Integer
164: function AnsiStrIComp(S1: PChar; S2: PChar): Integer)
165: Function AnsiStrLastChar( P : PChar) : PChar
166: function AnsiStrLastChar(P: PChar): PChar)
167: Function AnsiStrLComp( S1, S2 : PChar; MaxLen : Cardinal) : Integer
168: Function AnsiStrLComp( S1, S2 : PChar; MaxLen : Cardinal) : Integer
169: Function AnsiStrLower( Str : PChar) : PChar
170: Function AnsiStrPos( Str, SubStr : PChar) : PChar
171: function AnsiStrPos(Str: PChar; SubStr: PChar): PChar)
172: Function AnsiStrScan(Str: PChar; Chr: Char): PChar)
173: Function AnsiStrUpper( Str : PChar) : PChar
174: Function AnsiToUtf8( const S : string) : UTF8String
175: Function AnsiToUtf8Ex( const S : string; const cp : integer) : UTF8String
176: Function AnsiUpperCase( S : string) : string
177: Function AnsiUppercase(s : String) : String;
178: Function AnsiUpperCaseFileName( S : string) : string

```

```

179: Function ApplyUpdates(const Delta: OleVariant; MaxErrors: Integer; out ErrorCount: Integer): OleVariant
180: Function ApplyUpdates(const Delta: OleVariant; MaxErrors: Integer; out ErrorCount: Integer): OleVariant;
181: Function ApplyUpdates( MaxErrors : Integer) : Integer
182: Function ApplyUpdates1(const Delta: OleVar; MaxErrs: Int; out ErrCount: Int; var OwnerData: OleVar): OleVariant;
183: Function ArcCos( const X : Extended) : Extended
184: Function ArcCosh( const X : Extended) : Extended
185: Function ArcCot( const X : Extended) : Extended
186: Function ArcCotH( const X : Extended) : Extended
187: Function ArcCsc( const X : Extended) : Extended
188: Function ArcCscH( const X : Extended) : Extended
189: Function ArcSec( const X : Extended) : Extended
190: Function ArcSecH( const X : Extended) : Extended
191: Function ArcSin( const X : Extended) : Extended
192: Function ArcSinh( const X : Extended) : Extended
193: Function ArcTan( const X : Extended) : Extended
194: Function ArcTan2( const Y, X : Extended) : Extended
195: Function ArithmeticMean( const X : TDynDoubleArray) : Float
196: function ArrayLength: integer;
197: Function AsHex( const AValue : T4x4LongWordRecord) : string
198: Function AsHex( const AValue : T5x4LongWordRecord) : string
199: Function ASNDLen( var Start : Integer; const Buffer : string) : Integer
200: Function ASNDDecOIDItem( var Start : Integer; const Buffer : string) : Integer
201: Function ASNEncInt( Value : Integer) : string
202: Function ASNEncLen( Len : Integer) : string
203: Function ASNEncOIDItem( Value : Integer) : string
204: Function ASNEncUInt( Value : Integer) : string
205: Function ASNItem( var Start : Integer; const Buffer : string; var ValueType : Integer) : string
206: Function ASNObject( const Data : string; ASNTYPE : Integer) : string
207: Function Assigned(I: Longint): Boolean;
208: Function AspectRatio(aWidth, aHeight: Integer): String;
209: Function AsWideString( Field : TField) : WideString
210: Function AtLeast( ACount : Integer) : Boolean
211: Function AttemptToUseSharedMemoryManager : Boolean
212: Function Authenticate : Boolean
213: Function AuthenticateUser( const AUsername, APassword : String) : Boolean
214: Function Authentication : String
215: Function BatchMove( ASource : TBDEDataSet; AMode : TBatchMode) : Longint
216: Function BcdCompare( const bcd1, bcd2 : TBcd) : Integer
217: Function BcdFromBytes( const AValue : TBytes) : TBcd
218: Function BcdPrecision( const Bcd : TBcd) : Word
219: Function BcdScale( const Bcd : TBcd) : Word
220: Function BcdToBytes( const Value : TBcd) : TBytes
221: Function BCDToCurr( const BCD : TBcd; var Curr : Currency) : Boolean
222: Function BcdToDouble( const Bcd : TBcd) : Double
223: Function BcdToInteger( const Bcd : TBcd; Truncate : Boolean) : Integer
224: Function BcdToStr( const Bcd : TBcd) : string;
225: Function BcdToStrF(const Bcd : TBcd; Format: TFloatFormat; const Precision, Digits: Integer): string
226: function beep2(dwFreq, dwDuration: integer): boolean;
227: Function BeginPeriod( const Period : Cardinal) : Boolean
228: Function BeginTrans : Integer
229: Function BeginTransaction : TDBXTransaction;
230: Function BeginTransaction1( Isolation : TDBXIsolation) : TDBXTransaction;
231: function BigMulu(aone, atwo: string): string;
232: function BigNumber(aone, atwo: string): string;
233: function BigExp(aone, atwo: string): string;
234: function BigMul(aone, atwo: string): string;
235: function BigAdd(aone, atwo: string): string;
236: function BigSub(aone, atwo: string): string;
237: function BigFactorial(aone: string): string;
238: Function BinaryToDouble( ABinary : string; DefValue : Double) : Double
239: Function BinomialCoeff( N, R : Cardinal) : Float
240: function BinominalCoefficient(n, k: Integer): string;
241: Function BinStrToInt( const ABinary : String) : Integer
242: Function BinToByte(Binary: String): Byte;
243: function BinToHex2(Binary: String): string;
244: function BinToInt(Binary: String): Integer;
245: Function BinToChar(St: String): Char;
246: Function BinToStr(ans: string): string;
247: Function BitBlt(hdcDest:HDC;nXDest,nYDest,nWidth,nHeigh:Int;hdcSrc:HDC;nXSrc,nYSrc:Int;dwRop:DWORD):Bool;
248: Function BitmapsAreIdentical( ABitmap1, ABitmap2 : TBitmap) : Boolean
249: Function BitsHighest( X : Byte) : Integer;
250: Function BitsHighest1( X : ShortInt) : Integer;
251: Function BitsHighest2( X : SmallInt) : Integer;
252: Function BitsHighest3( X : Word) : Integer;
253: Function BitsHighest4( X : Integer) : Integer;
254: Function BitsHighest5( X : Cardinal) : Integer;
255: Function BitsHighest6( X : Int64) : Integer;
256: Function BitsLowest( X : Byte) : Integer;
257: Function BitsLowest1( X : Shortint) : Integer;
258: Function BitsLowest2( X : Smallint) : Integer;
259: Function BitsLowest3( X : Word) : Integer;
260: Function BitsLowest4( X : Cardinal) : Integer;
261: Function BitsLowest5( X : Integer) : Integer;
262: Function BitsLowest6( X : Int64) : Integer;
263: Function BitsNeeded( const X : Byte) : Integer;
264: Function BitsNeeded1( const X : Word) : Integer;
265: Function BitsNeeded2( const X : Integer) : Integer;
266: Function BitsNeeded3( const X : Int64) : Integer;
267: Function BlueComponent( const Color32 : TColor32) : Integer

```

```

268: Function BooleanToInteger( const Pb : Boolean) : Integer
269: Function BoolToStr(B: Boolean; UseBoolStrs: Boolean): string
270: Function BoolToStr1(value : boolean) : string;
271: Function booltoint( aBool : Boolean) : LongInt
272: Function inttobool( aInt : LongInt) : Boolean
273: Function Bounds( ALeft, ATop, AWidth, AHeight : Integer) : TRect
274: function Bounds(ALeft, ATop, AWidth, AHeight: Integer): TRect
275: Function BreakApart( BaseString, BreakString : string; StringList : TStringList) : TStringList
276: Function BrightColor( const Color : TColor; const Pct : Single) : TColor
277: Function BrightColorChannel( const Channel : Byte; const Pct : Single) : Byte
278: Function BufferRequest( Length : Integer) : TStream
279: Function BuildFileList( const Path : string; const Attr : Integer; const List : TStringList) : Boolean
280: Function Buttons : PTaskDialogButton
281: Function BytesPerScanline( PixelsPerScanline, BitsPerPixel, Alignment : Longint) : Longint
282: Function BytesToCardinal( const AValue : TIdBytes; const AIndex : Integer) : Cardinal
283: Function BytesToChar( const AValue : TIdBytes; const AIndex : Integer) : Char
284: Function BytesToInt64( const AValue : TIdBytes; const AIndex : Integer) : Int64
285: Function BytesToInteger( const AValue : TIdBytes; const AIndex : Integer) : Integer
286: Function BytesToIPv6( const AValue : TIdBytes; const AIndex : Integer) : TIdIPv6Address
287: Function BytesToShort( const AValue : TIdBytes; const AIndex : Integer) : Short
288: Function BytesToString( ABytes: TIdBytes; AStartIndex: Integer; AMaxCount: Integer): string;
289: Function BytesToStr( const Value: TBytes): String;
290: Function BytesToWord( const AValue : TIdBytes; const AIndex : Integer) : Word
291: Function ByteToBin( Int: Byte): String;
292: Function ByteToCharIndex( S : string; Index : Integer) : Integer
293: function ByteToCharIndex(const S: string; Index: Integer): Integer
294: Function ByteToCharLen( S : string; MaxLen : Integer) : Integer
295: function ByteToCharLen(const S: string; MaxLen: Integer): Integer
296: Function ByteToHex( const AByte : Byte) : string
297: Function ByteToOctal( const AByte : Byte) : string
298: Function ByteType( S : string; Index : Integer) : TmbcsByteType
299: function ByteType(const S: string; Index: Integer): TmbcsByteType
300: Function CalcTitleRect( Col : TColumn; ARow : Integer; var MasterCol : TColumn) : TRect
301: Function CalculateDFAFingerprint( oStates : TList) : integer
302: function CallTerminateProcs: Boolean
303: function CANFOCUS:BOOLEAN
304: Function CanLoad( const Ext : string) : Boolean
305: Function CanParse( AWebRequest : TWebRequest) : Boolean
306: Function CanSave( const Ext : string) : Boolean
307: Function CanStart( cChar : char) : boolean
308: Function CaptureScreen : TBitmap;
309: Function CaptureScreen1( Rec : TRect) : TBitmap;
310: Function CardinalToFourChar( ACardinal : LongWord) : string
311: Function CastSoapToNative(Info: PTypeInfo; const SoapData: WideString; NatData: Pointer; IsNull: Boolean): Boolean
312: Function CastSoapToVariant1( SoapInfo : PTypeInfo; const SoapData : WideString) : Variant;
313: Function Ceil( const X : Extended) : Integer
314: Function Ceil16( X : Integer) : Integer
315: Function Ceil4( X : Integer) : Integer
316: Function Ceil8( X : Integer) : Integer
317: Function Ceiling( const X : Extended) : Integer
318: Function CellRect( ACol, ARow : Longint) : TRect
319: Function CelsiusToFahrenheit( const AValue : Double) : Double
320: Function CenterPoint( const Rect : TRect) : TPoint
321: function CenterPoint(const Rect: TRect): TPoint
322: Function ChangeFileExt( FileName, Extension : string) : string
323: function ChangeFileExt(const FileName: string; const Extension: string): string
324: Function CharInSet2( const Ch : Char; const SetOfChar : TSetOfChar) : Boolean
325: Function CharInSet( const Ch : Char; const testSet : TSysCharSet) : Boolean
326: Function CharIsInEOF( const AString : string; ACharPos : Integer) : Boolean
327: Function CharIsInSet( const AString : string; const ACharPos : Integer; const ASet : String) : Boolean
328: Function CharLength( S : String; Index : Integer) : Integer
329: Function CharRange( const AMin, AMax : Char) : String
330: function CharSetToIdent(Charset: Longint; var Ident: string): Boolean
331: Function CharToBin(vChr: Char): String;
332: Function CharNext(lpsz: PChar): PChar; stdcall;
333: Function CharToByteIndex( S : string; Index : Integer) : Integer
334: function CharToByteIndex(const S: string; Index: Integer): Integer
335: Function CharToByteLen( S : string; MaxLen : Integer) : Integer
336: function CharToByteLen(const S: string; MaxLen: Integer): Integer
337: Function CharToHex(const APrefix : String; const cc : Char) : shortstring;
338: function CharToHexStr(Value: char): string;
339: function CharToOem(ins, outs: PChar):boolean;
340: function CharToUnicode(Value: Char): string;
341: Function CheckMenuDropdown : Boolean
342: Function CheckMessages : longint
343: Function CheckOpen( Status : DBIResult) : Boolean
344: Function CheckPassword( const APassword : String) : Boolean
345: Function CheckResponse(const AResponse: SmallInt; const AAllowedResponses: array of SmallInt): SmallInt
346: Function CheckCrc32( var X : array of Byte; N : Integer; Crc : Cardinal) : Integer;
347: function CheckSynchronize(Timeout: Integer): Boolean
348: Function CheckWin32Version( AMajor : Integer; AMinor : Integer) : Boolean
349: function ChrA(const a: byte): char;
350: Function ClassIDToProgID(const ClassID: TGUID): string;
351: Function ClassNameIs(const Name: string): Boolean
352: Function ClearBit( const Value : Byte; const Bit : TBitRange) : Byte;
353: Function ClearBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
354: Function ClearBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
355: Function ClearBit3( const Value : Word; const Bit : TBitRange) : Word;
356: Function ClearBit4( const Value : Integer; const Bit : TBitRange) : Integer;

```



```

357: Function ClearBit5( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
358: Function ClearBit6( const Value : Int64; const Bit : TBitRange) : Int64;
359: function CLIENTTOSCREEN(POINT:TPOINT):TPOINT
360: Function Clipboard : TClipboard
361: Function ClipCodes( const X, Y, MinX, MinY, MaxX, MaxY : Float) : TClipCodes;
362: Function ClipCodes1( const X, Y : Float; const ClipRect : TRect) : TClipCodes;
363: Function ClipLine( var X1, Y1, X2, Y2 : Integer; const ClipRect : TRect) : Boolean;
364: Function ClipLineToRect( var P1, P2 : TFloatPoint; const Rect : TFloatRect) : Boolean
365: Function Clone( out stm : IStream) : HRESULT
366: Function CloneConnection : TSQLConnection
367: Function CloneMemoryStream( Original : TMemoryStream) : TMemoryStream
368: function CLOSEQUERY:BOOLEAN
369: Function CloseVolume( var Volume : THandle) : Boolean
370: Function CloseHandle(Handle: Integer): Integer; stdcall;
371: Function CPLApplet( hwndCPL : THandle; uMsg : DWORD; lParam1, lParam2 : Longint) : Longint
372: Function CmdLine: PChar;
373: function CmdShow: Integer;
374: Function Color32( const R, G, B : Byte; const A : Byte) : TColor32;
375: Function Color32( WinColor : TColor) : TColor32;
376: Function Color321( const Index : Byte; const Palette : TPalette32) : TColor32;
377: Function ColorAdjustLuma( clrRGB : TColor; n : Integer; fScale : Boolean) : TColor
378: Function ColorHLSToRGB( Hue, Luminance, Saturation : Word) : TColorRef
379: Function ColorToHTML( const Color : TColor) : String
380: function ColorToIdent(Color: Longint; var Ident: string): Boolean)
381: Function ColorToRGB(color: TColor): Longint
382: function ColorToString(Color: TColor): string)
383: Function ColorToWebColorName( Color : TColor) : string
384: Function ColorToWebColorStr( Color : TColor) : string
385: Function ColumnAtDepth( Col : TColumn; ADepth : Integer) : TColumn
386: Function Combination(npr, ncr: integer): extended;
387: Function CombinationInt(npr, ncr: integer): Int64;
388: Function CombineInfo( Bitmap : TCustomBitmap32) : TCombineInfo
389: Function CommaAdd( const AStr1, AStr2 : String) : string
390: Function CommercialRound( const X : Extended) : Int64
391: Function Commit( grfCommitFlags : Longint) : HRESULT
392: Function Compare( const NameExt : string) : Boolean
393: function CompareDate(const A, B: TDateTime): TValueRelationship;
394: Function CompareDateTime( const ADateTime1, ADateTime2 : TDateTime) : Integer
395: Function CompareFiles(const FN1,FN2 :string; Breathe:TNotifyEvent;BreathingSender:TObject): boolean
396: Function CompareMemoryStreams( S1, S2 : TMemoryStream) : boolean
397: Function CompareStr( S1, S2 : string) : Integer
398: Function CompareStr(const S1: string; const S2: string): Integer)
399: function CompareString(const S1: string; const S2: string): Integer)
400: Function CompareText( S1, S2 : string) : Integer
401: function CompareText(const S1: string; const S2: string): Integer)
402: Function CompareTextLike(cWildStr,cStr:string;const cWildChar:char;lCaseSensitive:boolean): boolean
403: function CompareTime(const A, B: TDateTime): TValueRelationship;
404: Function CompatibleConversionType( const AType : TConvType; const AFamily : TConvFamily) : Boolean
405: Function CompatibleConversionTypes( const AFrom, ATo : TConvType) : Boolean
406: Function ComponentTypeToString( const ComponentType : DWORD) : string
407: Function CompToCurrency( Value : Comp) : Currency
408: Function CompToDouble( Value : Comp) : Double
409: function ComputeFileCRC32(const FileName : String) : Integer;
410: function ComputeSHA256(astr: string; amode: char): string) //mode F:File, S:String
411: function ComputeSHA512(astr: string; amode: char): string) //mode F:File, S:String
412: Function Concat(s: string): string
413: Function ConnectAndGetAll : string
414: Function Connected : Boolean
415: function constrain(x, a, b: integer): integer;
416: Function ConstraintCallBack( Req : DsInfoReq; var ADataSources : DataSources) : DBIResult
417: Function ConstraintsDisabled : Boolean
418: function CONTAINSCONTROL(CONTROL:TCONTROL):BOOLEAN
419: Function ContainsState( oState : TniRegularExpressionState) : boolean
420: Function ContainsStr( const AText, ASubText : string) : Boolean
421: Function ContainsText( const AText, ASubText : string) : Boolean
422: Function ContainsTransition( oTransition : TniRegularExpressionTransition) : boolean
423: Function Content : string
424: Function ContentFromStream( Stream : TStream) : string
425: Function ContentFromString( const S : string) : string
426: Function CONTROLSDISABLED : BOOLEAN
427: Function Convert( const AValue : Double; const AFrom, ATo : TConvType) : Double;
428: Function Convert1( const AValue : Double; const AFrom1, AFrom2, ATo1, ATo2 : TConvType) : Double;
429: Function ConvertFrom( const AFrom : TConvType; const AValue : Double) : Double
430: Function ConvertReadStream( Stream : TStream; Buffer : PChar; BufSize : Integer) : Integer
431: Function ConvertTo( const AValue : Double; const ATo : TConvType) : Double
432: Function ConvertWriteStream( Stream : TStream; Buffer : PChar; BufSize : Integer) : Integer
433: Function ConvFamilyToDescription( const AFamily : TConvFamily) : string
434: Function ConvTypeToDescription( const AType : TConvType) : string
435: Function ConvTypeToFamily( const AFrom, ATo : TConvType) : TConvFamily;
436: Function ConvTypeToFamily( const AType : TConvType) : TConvFamily;
437: Function ConvAdd(const AVal:Double;const AType1:TConvType;const AVal2:Double;const AType2,
AResultType:TConvType): Double
438: Function ConvCompareValue(const AValue1:Double;const AType1:TConvType;const AValue2:Double;const
AType2:TConvType): TValueRelationship
439: Function ConvDec( const AValue : Double; const AType, AAmountType : TConvType) : Double;
440: Function ConvDec1(const AValue:Double; const AType: TConvType;const AAmount:Double;const
AAmountType:TConvType):Double;
441: Function ConvDiff(const AVal1:Double;const AType1:TConvType;const AVal2:Double;const AType2,
AResultType:TConvType):Double

```

```

442: Function ConvInc( const AValue : Double; const AType, AAmountType : TConvType) : Double;
443: Function ConvIncl(const AValue:Double;const AType:TConvType;const AAmount:Double;const
      AAmountType:TConvType): Double;
444: Function ConvSameValue(const AValue1:Double;const AType1:TConvType;const AValue2:Double;const
      AType2:TConvType): Boolean
445: Function ConvToStr( const AValue : Double; const AType : TConvType) : string
446: Function ConvWithinNext( const AValue, ATest : Double; const AType : TConvType; const AAmount : Double;
      const AAmountType : TConvType) : Boolean
447: Function ConvWithinPrevious(const AValue,ATest:Double;const AType:TConvType; const AAmount:Double;const
      AAmountType: TConvType) : Boolean
448: Function Copy(s: AnyString; iFrom, iCount: Longint): AnyString;
449: Function CopyFile( Source, Dest : string; CanOverwrite : Boolean) : Boolean
450: Function CopyFileEx( Source, Dest : string; Flags : FILEOP_FLAGS) : Boolean
451: Function CopyFileTo( const Source, Destination : string) : Boolean
452: Function CopyFrom(Source:TStream;Count:Int64):LongInt
453: Function CopyPalette( Palette : HPALETTE) : HPALETTE
454: Function CopyTo( Length : Integer) : string
455: Function CopyTo(stm: IStream; cb: Largeint;out cbRead: Largeint;out cbWritten:Largeint): HRESULT
456: Function CopyToEOF : string
457: Function CopyToEOL : string
458: Function Cos(e : Extended) : Extended;
459: Function Cosecant( const X : Extended) : Extended
460: Function Cot( const X : Extended) : Extended
461: Function Cotan( const X : Extended) : Extended
462: Function CotH( const X : Extended) : Extended
463: Function Count : Integer
464: Function CountBitsCleared( X : Byte) : Integer;
465: Function CountBitsCleared1( X : Shortint) : Integer;
466: Function CountBitsCleared2( X : Smallint) : Integer;
467: Function CountBitsCleared3( X : Word) : Integer;
468: Function CountBitsCleared4( X : Integer) : Integer;
469: Function CountBitsCleared5( X : Cardinal) : Integer;
470: Function CountBitsCleared6( X : Int64) : Integer;
471: Function CountBitsSet( X : Byte) : Integer;
472: Function CountBitsSet1( X : Word) : Integer;
473: Function CountBitsSet2( X : Smallint) : Integer;
474: Function CountBitsSet3( X : ShortInt) : Integer;
475: Function CountBitsSet4( X : Integer) : Integer;
476: Function CountBitsSet5( X : Cardinal) : Integer;
477: Function CountBitsSet6( X : Int64) : Integer;
478: function CountGenerations(Ancestor,Descendent: TClass): Integer
479: Function Coversine( X : Float) : Float
480: function CRC32(const fileName: string): LongWord;
481: Function CREATEBLOBSTREAM( FIELD : TFIELD; MODE : TBLOBSTREAMMODE) : TSTREAM
482: Function CreateColumns : TDBGGridColumns
483: Function CreateDataLink : TGridDataLink
484: Function CreateDir( Dir : string) : Boolean
485: Function CreateDir(const Dir: string): Boolean
486: Function CreateDOSProcessRedirected( const CommandLine, InputFile, OutputFile : string) : Boolean
487: Function CreateEnvironmentBlock(const Options:TEnvironmentOptions;const AdditionalVars:TStrings): PChar
488: Function CREATEFIELD(OWNER:TCOMPONENT;PARENTFIELD:TOBJECTFIELD;const
      FIELDNAME:String;CREATECHILDREN:BOOLEAN):TFIELD
489: Function CreateGlobber( sFilespec : string) : TniRegularExpression
490: Function CreateGrayMappedBmp( Handle : HBITMAP) : HBITMAP
491: Function CreateGrayMappedRes( Instance : THandle; ResName : PChar) : HBITMAP
492: function CreateGUID(out Guid: TGUID): HRESULT)
493: Function CreateInstance( const unkOuter : IUnknown; const iid : TGUID; out obj ) : HRESULT
494: Function CreateMappedBmp( Handle : HBITMAP; const OldColors, NewColors : array of TColor) : HBITMAP
495: Function CreateMappedRes(Instance:THandle;ResName:PChar;const OldColors,NewColors:array of TColor):HBITMAP
496: Function CreateMessageDialog(const Msg:string; DlgType:TMsgDlgType; Buttons: TMsgDlgButtons) : TForm;
497: Function CreateMessageDialog1(const
      Msg:string;DlgType:TMsgDlgType;Btns:TMsgDlgBtns;DefaultBtn:TMsgDlgBtn):TForm;
498: function CreateOleObject(const ClassName: String): IDispatch;
499: Function CREATEPARAM( FLDTYPE : TFIELDTYPE; const PARAMNAME : String; PARAMTYPE : TPARAMTYPE) : TPARAM
500: Function CreateParameter(const
      Name:WideString;DataType:TDataType;Direction:TParameterDirection;Size:Integer;Value: OleVariant):TParameter
501: Function CreateLocate( DataSet : TDataSet) : TJvLocateObject
502: Function CreateMappedBmp( Handle : HBITMAP; const OldColors, NewColors : array of TColor) : HBITMAP
503: Function CreateMappedRes(Instance:THandle;ResName:PChar;const OldColors,NewColors:array of TColor):HBITMAP
504: Function CreateRecordBuffer( Length : Integer) : TRecordBuffer
505: Function CreateValueBuffer( Length : Integer) : TValueBuffer
506: Function CreatePopUpCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode) : TWinControl
507: Function CreateRecordBuffer( Length : Integer) : TRecordBuffer
508: Function CreateRotatedFont( Font : TFont; Angle : Integer) : HFONT
509: Function CreateTwoColorsBrushPattern( Color1, Color2 : TColor) : TBitmap
510: Function CreateValueBuffer( Length : Integer) : TValueBuffer
511: Function CreateHexDump( AOwner : TWinControl) : THexDump
512: Function Csc( const X : Extended) : Extended
513: Function CscH( const X : Extended) : Extended
514: function currencyDecimals: Byte
515: function currencyFormat: Byte
516: function currencyString: String
517: Function CurrentProcessId : TIdPID
518: Function CurrentReadBuffer : string
519: Function CurrentThreadId : TIdPID
520: Function CurrentYear : Word
521: Function CurrToBCD(const Curr: Currency; var BCD: TBcd; Precision: Integer; Decimals: Integer): Boolean
522: Function CurrToStr( Value : Currency) : string;
523: Function CurrToStrF( Value : Currency; FormatSettings : TFormatSettings; Digits: Integer) : string;

```

```

524: Function CurrToStrFS(Value:Currency;Format:TFloatFormat;Digits:Integer;const
    FormatSettings:TFormatSettings):string;
525: function CursorToIdent(cursor: Longint; var Ident: string): Boolean;
526: function CursorToString(cursor: TCursor): string;
527: Function CustomSort( SortProc : TLVCompare; lParam : Longint) : Boolean
528: Function CustomSort( SortProc : TTVCompare; Data : Longint; ARecurse : Boolean) : Boolean
529: Function CycleToDeg( const Cycles : Extended) : Extended
530: Function CycleToGrad( const Cycles : Extended) : Extended
531: Function CycleToRad( const Cycles : Extended) : Extended
532: Function D2H( N : Longint; A : Byte) : string
533: Function DarkColor( const Color : TColor; const Pct : Single) : TColor
534: Function DarkColorChannel( const Channel : Byte; const Pct : Single) : Byte
535: Function DataLinkDir : string
536: Function DataRequest( Data : OleVariant) : OleVariant
537: Function DataRequest( Input : OleVariant) : OleVariant
538: Function DataToRawColumn( ACol : Integer) : Integer
539: Function Date : TDateTime
540: function Date: TDateTime;
541: Function DateIsNull( const pdtValue : TDateTime; const pdtKind : TdtKind) : Boolean
542: Function DateOf( const AValue : TDateTime) : TDateTime
543: function DateSeparator: char;
544: Function DateTimeGMTToHttpStr( const GMTValue : TDateTime) : String
545: Function DateTimeToFileDate( DateTime : TDateTime) : Integer
546: function DateTimeToFileDate(DateTime: TDateTime): Integer;
547: Function DateTimeToGmtOffsetStr( ADateTime : TDateTime; SubGMT : Boolean) : string
548: Function DateTimeToInternetStr( const Value : TDateTime; const AIsGMT : Boolean) : String
549: Function DateTimeToJulianDate( const AValue : TDateTime) : Double
550: Function DateTimeToModifiedJulianDate( const AValue : TDateTime) : Double
551: Function DateTimeToStr( DateTime : TDateTime) : string;
552: Function DateTimeToStr2( DateTime : TDateTime; FormatSettings : TFormatSettings) : string;
553: function DateTimeToTimeStamp(DateTime: TDateTime): TTimeStamp
554: Function DateTimeToUnix( const AValue : TDateTime) : Int64
555: function DateTimeToUnix(D: TDateTime): Int64;
556: Function DateToStr( DateTime : TDateTime) : string;
557: function DateToStr(const DateTime: TDateTime): string;
558: function DateToStr(D: TDateTime): string;
559: Function DateToStr2( DateTime : TDateTime; FormatSettings : TFormatSettings) : string;
560: Function DayOf( const AValue : TDateTime) : Word
561: Function DayOfTheMonth( const AValue : TDateTime) : Word
562: function DayOfTheMonth(const AValue: TDateTime): Word;
563: Function DayOfTheWeek( const AValue : TDateTime) : Word
564: Function DayOfTheYear( const AValue : TDateTime) : Word
565: function DayOfTheYear(const AValue: TDateTime): Word;
566: Function DayOfWeek( DateTime : TDateTime) : Word
567: function DayOfWeek(const DateTime: TDateTime): Word;
568: Function DayOfWeekStr( DateTime : TDateTime) : string
569: Function DaysBetween( const ANow, ATen : TDateTime) : Integer
570: Function DaysInAMonth( const AYear, AMonth : Word) : Word
571: Function DaysInAYear( const AYear : Word) : Word
572: Function DaysInMonth( const AValue : TDateTime) : Word
573: Function DaysInYear( const AValue : TDateTime) : Word
574: Function DaySpan( const ANow, ATen : TDateTime) : Double
575: Function DBUseRightToLeftAlignment( AControl : TControl; AField : TField) : Boolean
576: function DecimalSeparator: char;
577: Function DecLimit( var B : Byte; const Limit : Byte; const Decr : Byte) : Byte;
578: Function DecLimit1( var B : Shortint; const Limit : Shortint; const Decr : Shortint) : Shortint;
579: Function DecLimit2( var B : Smallint; const Limit : Smallint; const Decr : Smallint) : Smallint;
580: Function DecLimit3( var B : Word; const Limit : Word; const Decr : Word) : Word;
581: Function DecLimit4( var B : Integer; const Limit : Integer; const Decr : Integer) : Integer;
582: Function DecLimit5( var B : Cardinal; const Limit : Cardinal; const Decr : Cardinal) : Cardinal;
583: Function DecLimit6( var B : Int64; const Limit : Int64; const Decr : Int64) : Int64;
584: Function DecLimitClamp( var B : Byte; const Limit : Byte; const Decr : Byte) : Byte;
585: Function DecLimitClamp1( var B : Shortint; const Limit : Shortint; const Decr : Shortint) : Shortint;
586: Function DecLimitClamp2( var B : Smallint; const Limit : Smallint; const Decr : Smallint) : Smallint;
587: Function DecLimitClamp3( var B : Word; const Limit : Word; const Decr : Word) : Word;
588: Function DecLimitClamp4( var B : Integer; const Limit : Integer; const Decr : Integer) : Integer;
589: Function DecLimitClamp5( var B : Cardinal; const Limit : Cardinal; const Decr : Cardinal) : Cardinal;
590: Function DecLimitClamp6( var B : Int64; const Limit : Int64; const Decr : Int64) : Int64;
591: Function DecodeDateFully( DateTime : TDateTime; var Year, Month, Day, DOW : Word) : Boolean
592: Function DecodeSoundexInt( AValue : Integer) : string
593: Function DecodeSoundexWord( AValue : Word) : string
594: Function DefaultAlignment : TAlignment
595: Function DefaultCaption : string
596: Function DefaultColor : TColor
597: Function DefaultFont : TFont
598: Function DefaultImeMode : TImeMode
599: Function DefaultImeName : TImeName
600: Function DefaultReadOnly : Boolean
601: Function DefaultWidth : Integer
602: Function DegMinSecToFloat( const Degs, Mins, Secs : Float) : Float
603: Function DegToCycle( const Degrees : Extended) : Extended
604: Function DegToGrad( const Degrees : Extended) : Extended
605: Function DegToGrad( const Value : Extended) : Extended;
606: Function DegToGrad1( const Value : Double) : Double;
607: Function DegToGrad2( const Value : Single) : Single;
608: Function DegToRad( const Degrees : Extended) : Extended
609: Function DegToRad( const Value : Extended) : Extended;
610: Function DegToRad1( const Value : Double) : Double;
611: Function DegToRad2( const Value : Single) : Single;

```

```

612: Function DelChar( const pStr : string; const pChar : Char) : string
613: Function DelEnvironmentVar( const Name : string) : Boolean
614: Function Delete( const MsgNum : Integer) : Boolean
615: Function DeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean) : Boolean
616: Function DeleteFile(const FileName: string): boolean
617: Function DeleteFileEx( FileName : string; Flags : FILEOP_FLAGS) : Boolean
618: Function DelimiterPosn( const sString : string; const sDelimiters: string): integer;
619: Function DelimiterPosn1(const sString:string;const sDelimiters:string;out cDelimiter: char) : integer;
620: Function DelSpace( const pStr : string) : string
621: Function DelString( const pStr, pDelStr : string) : string
622: Function DelTree( const Path : string) : Boolean
623: Function Depth : Integer
624: Function Description : string
625: Function DescriptionsAvailable : Boolean
626: Function DescriptionToConvFamily( const ADescription : string; out AFamily : TConvFamily) : Boolean
627: Function DescriptionToConvType( const ADescription : string; out AType : TConvType) : Boolean;
628: Function DescriptionToConvType1(const AFamil:TConvFamily;const ADescr:string;out AType:TConvType):Boolean;
629: Function DetectUTF8Encoding( const s : UTF8String) : TEncodeType
630: Function DialogsToPixelsX( const Dialogs : Word) : Word
631: Function DialogsToPixelsY( const Dialogs : Word) : Word
632: Function Digits( const X : Cardinal) : Integer
633: Function DirectoryExists( const Name : string) : Boolean
634: Function DirectoryExists( Directory : string) : Boolean
635: Function DiskFree( Drive : Byte) : Int64
636: function DiskFree(Drive: Byte): Int64
637: Function DiskInDrive( Drive : Char) : Boolean
638: Function DiskSize( Drive : Byte) : Int64
639: function DiskSize(Drive: Byte): Int64
640: Function DISPATCHCOMMAND( ACOMMAND : WORD) : BOOLEAN
641: Function DispatchEnabled : Boolean
642: Function DispatchMask : TMask
643: Function DispatchMethodType : TMethodType
644: Function DISPATCHPOPUP( AHANDLE : HMENU) : BOOLEAN
645: Function DispatchRequest( Sender : TObject; Request : TWebRequest; Response : TWebResponse) : Boolean
646: Function DisplayCase( const S : String) : String
647: Function DisplayRect( Code : TDisplayCode) : TRect
648: Function DisplayRect( TextOnly : Boolean) : TRect
649: Function DisplayStream( Stream : TStream) : string
650: TBufferCoord', 'record Char : integer; Line : integer; end
651: TDisplayCoord', 'record Column : integer; Row : integer; end
652: Function DisplayCoord( AColumn, ARow : Integer) : TDisplayCoord
653: Function BufferCoord( AChar, ALine : Integer) : TBufferCoord
654: Function DomainName( const AHost : String) : String
655: Function DosPathToUnixPath( const Path : string) : string
656: Function DottedLineTo( const Canvas : TCanvas; const X, Y : Integer) : Boolean;
657: Function DoubleDecliningBalance( const Cost, Salvage : Extended; Life, Period : Integer) : Extended
658: Function DoubleToBcd( const AValue : Double) : TBcd;
659: Function DoubleToHex( const D : Double) : string
660: Function DoUpdates : Boolean
661: function Dragging: Boolean;
662: Function DrawCaption( p1 : HWND; p2 : HDC; const p3 : TRect; p4 : UINT) : BOOL
663: Function DrawAnimatedRects( hwnd : HWND; idAni : Integer; const lprcFrom, lprcTo : TRect) : BOOL
664: Function DrawEdge( hdc : HDC; var qrc : TRect; edge : UINT; grfFlags : UINT) : BOOL
665: Function DrawFrameControl( DC : HDC; const Rect : TRect; uType, uState : UINT) : BOOL
666: {Works like InputQuery but displays 2edits. If PasswordChar <> #0, the second edit's PasswordChar is set}
667: Function DualInputQuery(const ACapt,Prpt1,Prpt2:string;var AVall,AVal2:string;PasswrdrChar:Char= #0):Bool;
668: Function DupeString( const AText : string; ACount : Integer) : string
669: Function Edit : Boolean
670: Function EditCaption : Boolean
671: Function EditText : Boolean
672: Function EditFolderList( Folders : TStrings) : Boolean
673: Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext) : Boolean
674: Function Elapsed( const Update : Boolean) : Cardinal
675: Function EnableProcessPrivilege( const Enable : Boolean; const Privilege : string) : Boolean
676: Function EnableThreadPrivilege( const Enable : Boolean; const Privilege : string) : Boolean
677: Function EncodeDate( Year, Month, Day : Word) : TDateTime
678: Function EncodeDate(Year, Month, Day: Word): TDateTime;
679: Function EncodeDateDay( const AYear, ADayOfYear : Word) : TDateTime
680: Function EncodeDateMonthWeek( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word) : TDateTime
681: Function EncodeDateTime(const AYear,AMonth,ADay,AMinute,ASecond,AMilliSecond: Word): TDateTime
682: Function EncodeDateWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
683: Function EncodeDayOfWeekInMonth( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word) : TDateTime
684: Function EncodeString( s : string) : string
685: Function DecodeString( s : string) : string
686: Function EncodeTime( Hour, Min, Sec, MSec : Word) : TDateTime
687: Function EncodeTime(Hour, Min, Sec, MSec: Word): TDateTime;
688: Function EndIP : String
689: Function EndOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
690: Function EndOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
691: Function EndOfAMonth( const AYear, AMonth : Word) : TDateTime
692: Function EndOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
693: Function EndOfAYear( const AYear : Word) : TDateTime
694: Function EndOfTheDay( const AValue : TDateTime) : TDateTime
695: Function EndOfTheMonth( const AValue : TDateTime) : TDateTime
696: Function EndOfTheWeek( const AValue : TDateTime) : TDateTime
697: Function EndOfTheYear( const AValue : TDateTime) : TDateTime
698: Function EndPeriod( const Period : Cardinal) : Boolean
699: Function EndsStr( const ASubText, AText : string) : Boolean
700: Function EndsText( const ASubText, AText : string) : Boolean

```



```

701: Function EnsureMsgIDBrackets( const AMsgID : String ) : String
702: Function EnsureRange( const AValue, AMin, AMax : Integer ) : Integer;
703: Function EnsureRange1( const AValue, AMin, AMax : Int64 ) : Int64;
704: Function EnsureRange2( const AValue, AMin, AMax : Double ) : Double;
705: Function EOF: boolean
706: Function EOLn: boolean
707: Function EqualRect( const R1, R2 : TRect ) : Boolean
708: function EqualRect(const R1, R2: TRect): Boolean)
709: Function Equals( Strings : TWideStrings ) : Boolean
710: function Equals(Strings: TStrings): Boolean;
711: Function EqualState( oState : TniRegularExpressionState ) : boolean
712: Function ErrOutput: Text)
713: function ExceptionParam: String;
714: function ExceptionPos: Cardinal;
715: function ExceptionProc: Cardinal;
716: function ExceptionToString(er: TIFException; Param: String): String;
717: function ExceptionType: TIFException;
718: Function ExcludeTrailingBackslash( S : string ) : string
719: function ExcludeTrailingBackslash(const S: string): string)
720: Function ExcludeTrailingPathDelimiter( const APath : string ) : string
721: Function ExcludeTrailingPathDelimiter( S : string ) : string
722: function ExcludeTrailingPathDelimiter(const S: string): string)
723: function ExecConsoleApp(const AppName, Parameters: String; AppOutput: TStrings): boolean;
724: Function ExecProc : Integer
725: Function ExecSQL : Integer
726: Function ExecSQL( ExecDirect : Boolean ) : Integer
727: Function Execute : _Recordset;
728: Function Execute : Boolean
729: Function Execute : Boolean;
730: Function Execute( const SQL : string; Params : TParams; Cache : Boolean; cursor : phDBCur ) : Integer
731: Function Execute( const SQL : WideString; Params : TParams; ResultSet : TPSResult ) : Integer
732: Function Execute( ParentWnd : HWND ) : Boolean
733: Function Executel(constCommText:WideString;const CType:TCommType;const
ExecuteOpts:TExecuteOpts):_Recordset;
734: Function Executel( const Parameters : OleVariant ) : _Recordset;
735: Function Executel( ParentWnd : HWND ) : Boolean;
736: Function Execute2( var RecordsAffected : Integer; const Parameters : OleVariant ) : _Recordset;
737: Function ExecuteAction( Action : TBasicAction ) : Boolean
738: Function ExecuteDirect( const SQL : WideString ) : Integer
739: Function ExecuteFile(const FileName:string;const Params:string;const DefDir:string;ShowCmd:Int):THandle
740: Procedure ExecuteThread2(afunc:TThreadFunction2;thrOK:boolean);AddTypeS('TThreadFunction2','procedure
741: Function CreateThread2(ThreadFunc: TThreadFunction2) : THandle
742: function ExeFileIsRunning(ExeFile: string): boolean;
743: function ExePath: string;
744: function ExePathName: string;
745: Function Exists( AItem : Pointer ) : Boolean
746: Function ExitWindows( ExitCode : Cardinal ) : Boolean
747: function Exp(x: Extended): Extended;
748: Function ExpandEnvironmentVar( var Value : string ) : Boolean
749: Function ExpandFileName( FileName : string ) : string
750: function ExpandFileName(const FileName: string): string)
751: Function ExpandUNCFileName( FileName : string ) : string
752: function ExpandUNCFileName(const FileName: string): string)
753: Function ExpJ( const X : Float ) : Float;
754: Function Exsecans( X : Float ) : Float
755: Function Extract( const AByteCount : Integer ) : string
756: Function Extract( Item : TClass ) : TClass
757: Function Extract( Item : TComponent ) : TComponent
758: Function Extract( Item : TObject ) : TObject
759: Function ExtractFileDir( FileName : string ) : string
760: function ExtractFileDir(const FileName: string): string)
761: Function ExtractFileDrive( FileName : string ) : string
762: function ExtractFileDrive(const FileName: string): string)
763: Function ExtractFileExt( FileName : string ) : string
764: function ExtractFileExt(const FileName: string): string)
765: Function ExtractFileExtNoDot( const FileName : string ) : string
766: Function ExtractFileExtNoDotUpper( const FileName : string ) : string
767: Function ExtractFileName( FileName : string ) : string
768: function ExtractFileName(const filename: string):string;
769: Function ExtractFilePath( FileName : string ) : string
770: function ExtractFilePath(const filename: string):string;
771: Function ExtractRelativePath( BaseName, DestName : string ) : string
772: function ExtractRelativePath(const BaseName: string; const DestName: string): string)
773: Function ExtractShortPathName( FileName : string ) : string
774: function ExtractShortPathName(const FileName: string): string)
775: Function ExtractStrings( Separators, WhiteSpace: TSysCharSet; Content: PChar; Strings: TStrings): Integer
776: function ExtractStrings(Separators:TSysCharSet;WhiteSpace:TSysCharSet;Content:PChar;Str:TStrings): Integer)
777: Function Fact(num: integer): Extended;
778: Function FactInt(num: integer): int64;
779: Function Factorial( const N : Integer ) : Extended
780: Function FahrenheitToCelsius( const AValue : Double ) : Double
781: function FalseBoolStrs: array of string
782: Function Fetch(var AInput:string;const ADelim:string;const ADelete:Bool;const ACaseSensitive:Bool):string
783: Function FetchCaseInsensitive(var AInput:string; const ADelim:string; const ADelete:Boolean): string
784: Function Fibo(num: integer): Extended;
785: Function FiboInt(num: integer): Int64;
786: Function Fibonacci( const N : Integer ) : Integer
787: Function FIELDBYNAME( const FIELDNAME : STRING ) : TFIELD
788: Function FIELDBYNAME( const FIELDNAME : WIDESTRING ) : TFIELD

```

```

789: Function FIELDBYNAME( const NAME : String ) : TFIELD
790: Function FIELDBYNAME( const NAME : String ) : TFIELDDEF
791: Function FIELDBYNUMBER( FIELDNO : Integer ) : TFIELD
792: Function FileAge( FileName : string ) : Integer
793: Function FileAge(const FileName: string): integer
794: Function FileCompareText( const A, B : String ) : Integer
795: Function FileContains(const FileName:string;Text:string;CaseSensitive:Bool;ExceptionOnError:Bool): Boolean
796: Function FileCreate( FileName : string ) : Integer;
797: Function FileCreate(const FileName: string): integer
798: Function FileCreateTemp( var Prefix : string ) : THandle
799: Function FileDateTimeToDate( FileDate : Integer ) : TDateTime
800: function FileDateTimeToDate(FileDate: Integer): TDateTime;
801: Function FileExists( const FileName : string ) : Boolean
802: Function FileExists( FileName : string ) : Boolean
803: function fileExists(const FileName: string): Boolean;
804: Function FileGetAttr( FileName : string ) : Integer
805: Function FileGetAttr(const FileName: string): integer
806: Function FileGetDate( Handle : Integer ) : Integer
807: Function FileGetDate(handle: integer): integer
808: Function FileGetDisplayName( const FileName : string ) : string
809: Function FileGetSize( const FileName : string ) : Integer
810: Function FileGetTempName( const Prefix : string ) : string
811: Function FileGetTypeNames( const FileName : string ) : string
812: Function FileIsReadOnly( FileName : string ) : Boolean
813: Function FileLoad( ResType : TResType; const Name : string; MaskColor : TColor ) : Boolean
814: Function FileOpen( FileName : string; Mode : LongWord ) : Integer
815: Function FileOpen(const FileName: string; mode:integer): integer
816: Function FileRead(handle: integer; Buffer: PChar; count: LongWord): integer
817: Function FileSearch( Name, DirList : string ) : string
818: Function FileSearch(const Name, dirList: string): string
819: Function FileSeek( Handle : Integer; Offset : Int64; Origin : Integer ) : Int64;
820: Function FileSeek( Handle, Offset, Origin : Integer ) : Integer;
821: Function FileSeek(handle, offset, origin: integer): integer
822: Function FileSetAttr( FileName : string; Attr : Integer ) : Integer
823: function FileSetAttr(const FileName: string; Attr: Integer): Integer
824: Function FileSetDate(FileName : string; Age : Integer) : Integer;
825: Function FileSetDate(handle: integer; age: integer): integer
826: Function FileSetDate2(FileHandle : Integer; Age : Integer) : Integer;
827: Function FileSetDateH( Handle : Integer; Age : Integer ) : Integer;
828: Function FileSetReadOnly( FileName : string; ReadOnly : Boolean ) : Boolean
829: Function FileSize( const FileName : string ) : int64
830: Function FileSizeByName( const AFilename : string ) : Longint
831: function FileWrite(Handle: Integer; const Buffer: pChar; Count: LongWord): Integer
832: Function FilterSpecArray : TComdlgFilterSpecArray
833: Function FIND( ACAPTION : String ) : TMenuItem
834: Function Find( AItem : Pointer; out AData : Pointer ) : Boolean
835: Function FIND( const ANAME : String ) : TMenuItem
836: Function Find( const DisplayName : string ) : TAggregate
837: Function Find( const Item : TBookmarkStr; var Index : Integer ) : Boolean
838: Function FIND( const NAME : String ) : TFIELD
839: Function FIND( const NAME : String ) : TFIELDDEF
840: Function FIND( const NAME : String ) : TINDEXDEF
841: Function Find( const S : WideString; var Index : Integer ) : Boolean
842: function Find(S:String;var Index:Integer):Boolean
843: Function FindAuthClass( AuthName : String ) : TIdAuthenticationClass
844: Function FindBand( AControl : TControl ) : TColorBand
845: Function FindBoundary( AContentType : string ) : string
846: Function FindButton( AModalResult : TModalResult ) : TTaskDialogBaseButtonItem
847: Function FindCaption(StartIndex: Integer;Value: string; Partial,Inclusive,Wrap: Boolean): TListItem
848: Function FindCdLineSwitch( Switch : string; IgnoreCase : Boolean ) : Boolean;
849: Function FindCloseW(FindFile: Integer): LongBool; stdcall;
850: Function FindCmdLineSwitch( Switch : string; Chars : TSysCharSet; IgnoreCase : Boolean ) : Boolean;
851: Function FindCmdLineSwitch( Switch : string ) : Boolean;
852: function FindComponent(ANAME: String): TComponent;
853: function FindComponent(vlabel: string): TComponent;
854: function FindComponent2(vlabel: string): TComponent;
855: function FindControl(Handle: HWND): TWinControl;
856: Function FindData( StartIndex : Integer; Value : Pointer; Inclusive, Wrap : Boolean ) : TListItem
857: Function FindDatabase( const DatabaseName : string ) : TDatabase
858: function FindDragTarget(const Pos: TPoint; AllowDisabled: Boolean): TControl;
859: Function FINDFIELD( const FIELDNAME : STRING ) : TFIELD
860: Function FINDFIELD( const FIELDNAME : WideString ) : TFIELD
861: Function FindFirst2(const Path: string; Attr: Integer; var F: TSearchRec):Integer
862: Function FindNext2(var F: TSearchRec): Integer
863: procedure FindClose2(var F: TSearchRec)
864: Function FINDFIRST : BOOLEAN
865: TJvSpecialFolder = (sfRecycleBin, sfControlPanel, sfDesktop, sfDesktopDirectory,
866: sfMyComputer, sfFonts, sfNetHood, sfNetwork, sfPersonal, sfPrinters, sfPrograms,sfRecent,sfSendTo,
sfStartMenu, sfStartup, sfTemplates);
867: FFolder: array [TJvSpecialFolder] of Integer =
868: (CSIDL_BITBUCKET, CSIDL_CONTROLS, CSIDL_DESKTOP, CSIDL_DESKTOPDIRECTORY,
869: CSIDL_DRIVES, CSIDL_FONTS, CSIDL_NETHOOD, CSIDL_NETWORK, CSIDL_PERSONAL,
870: CSIDL_PRINTERS, CSIDL_PROGRAMS, CSIDL_RECENT, CSIDL_SENDTO, CSIDL_STARTMENU,
871: CSIDL_STARTUP, CSIDL_TEMPLATES);
872: Function FindFilesDlg(StartIn: string; SpecialFolder: TJvSpecialFolder; UseFolder: Boolean): Boolean);
873: function Findfirst(const filepath: string; attr: integer): integer;
874: function FindFirst2(const Path: string; Attr: Integer; var F: TSearchRec): Integer
875: Function FindFirstNotOf( AFind, AText : String ) : Integer
876: Function FindFirstOf( AFind, AText : String ) : Integer

```

```

877: Function FindImmediateTransitionOn( cChar : char ) : TniRegularExpressionState
878: Function FINDINDEXFORFIELDS( const FIELDS : String ) : TINDEXDEF
879: Function FindInstanceOf( AClass : TClass; AExact : Boolean; AStartAt : Integer ) : Integer
880: Function FINDITEM( VALUE : INTEGER; KIND : TFINDITEMKIND ) : TMENUITEM
881: Function FindItemId( Id : Integer ) : TCollectionItem
882: Function FindKey( const KeyValues : array of const ) : Boolean
883: Function FINDLAST : BOOLEAN
884: Function FindLineControl( ComponentType, ControlType : DWORD ) : TJclMixerLineControl
885: Function FindModuleClass( AClass : TComponentClass ) : TComponent
886: Function FindModuleName( const AClass : string ) : TComponent
887: Function FINDNEXT : BOOLEAN
888: function FindNext: integer;
889: function FindNext2(var F: TSearchRec): Integer)
890: Function FindNextPage( CurPage : TTabSheet; GoForward, CheckTabVisible : Boolean ) : TTabSheet
891: Function FindNextToSelect : TTreeNode
892: Function FINDPARAM( const VALUE : String ) : TPARAM
893: Function FindParam( const Value : WideString ) : TParameter
894: Function FINDPRIOR : BOOLEAN
895: Function FindResource( ModuleHandle : HMODULE; ResourceName, ResourceType : PChar ) : TResourceHandle
896: Function FindSession( const SessionName : string ) : TSession
897: function FindStringResource(Ident: Integer): string)
898: Function FindText( const SearchStr:string;StartPos,Length: Integer; Options: TSearchTypes):Integer
899: Function FindUnusedFileName( const FileName, FileExt, Suffix : AnsiString ) : AnsiString
900: function FindVCLWindow(const Pos: TPoint): TWinControl;
901: function FindWindow(C1, C2: PChar): Longint;
902: Function FindInPaths(const fileName,paths: String): String;
903: Function Finger : String
904: Function First : TClass
905: Function First : TComponent
906: Function First : TObject
907: Function FirstDelimiter( const delimiters : string; const Str : String ) : integer;
908: Function FirstDelimiter1( const delimiters : WideString; const Str : WideString ) : integer;
909: Function FirstInstance( const ATitle : string ) : Boolean
910: Function FloatPoint( const X, Y : Float ) : TFloatPoint;
911: Function FloatPoint1( const P : TPoint ) : TFloatPoint;
912: Function FloatPtInRect( const Rect : TFloatRect; const P : TFloatPoint ) : Boolean
913: Function FloatRect( const ALeft, ATop, ARight, ABottom : Double ) : TFloatRect;
914: Function FloatRect1( const Rect : TRect ) : TFloatRect;
915: Function FloatsEqual( const X, Y : Float ) : Boolean
916: Function FloatToBin(const D: Double): string; //doubletohex -> hexto bin! in buffer
917: Function FloatToCurr( Value : Extended ) : Currency
918: Function FloatToDateTime( Value : Extended ) : TDateTime
919: Function FloatToStr( Value : Extended ) : string;
920: Function FloatToStr(e : Extended) : string;
921: Function FloatToStrF( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer ) : string;
922: function FloatToStrF(Value: Extended; Format: TFloatFormat; Precision: Integer; Digits: Integer): string)
923: Function FloatToStr2( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer; FormatSettings : TFormatSettings ) : string;
924: Function FloatToStrFS( Value : Extended; Format : TFloatFormat; Precision, Digits : Integer; FormatSettings : TFormatSettings ) : string;
925: function FloatToText(BufferArg: PChar; const Value: Extended; ValueType: TFloatValue; Format: TFloatFormat; Precision,Digits: Integer): Integer)
926: Function Floor( const X : Extended ) : Integer
927: Function FloorInt( Value : Integer; StepSize : Integer ) : Integer
928: Function FloorJ( const X : Extended ) : Integer
929: Function Flush( const Count : Cardinal ) : Boolean
930: Function Flush(var t: Text): Integer
931: function FmtLoadStr(Ident: Integer; const Args: array of const): string)
932: function FOCUSED:BOOLEAN
933: Function ForceBackslash( const PathName : string ) : string
934: Function ForceDirectories( const Dir : string ) : Boolean
935: Function ForceDirectories( Dir : string ) : Boolean
936: Function ForceDirectories( Name : string ) : Boolean
937: Function ForceInBox( const Point : TPoint; const Box : TRect ) : TPoint
938: Function ForceInRange( A, Min, Max : Integer ) : Integer
939: Function ForceInRangeR( const A, Min, Max : Double ) : Double
940: Function ForEach( AProc : TBucketProc; AInfo : Pointer ) : Boolean;
941: Function ForEach1( AEvent : TBucketEvent ) : Boolean;
942: Function ForegroundTask: Boolean
943: function Format(const Format: string; const Args: array of const): string;
944: Function FormatBcd( const Format : string; Bcd : TBcd ) : string
945: FUNCTION FormatBigInt(s: string): STRING;
946: function FormatBuf(var Buffer:PChar;BufLen:Card;const Format:string;FmtLen:Cardinal;const Args:array of const):Cardinal
947: Function FormatCount( iCount : integer; const sSingular : string; const sPlural : string ) : string
948: Function FormatCurr( Format : string; Value : Currency ) : string;
949: function FormatCurr(const Format: string; Value: Currency): string)
950: Function FormatDateTime( Format : string; DateTime : TDateTime ) : string;
951: Function FormatDateTime(const fmt: string; D: TDateTime): string;
952: Function FormatFloat( Format : string; Value : Extended ) : string;
953: function FormatFloat(const Format: string; Value: Extended): string)
954: Function FormatFloat( Format : string; Value : Extended ) : string;
955: Function FormatFloat2( Format : string; Value : Extended; FormatSettings : TFormatSettings ) : string;
956: Function FormatCurr( Format : string; Value : Currency ) : string;
957: Function FormatCurr2(Format: string; Value : Currency; FormatSettings : TFormatSettings ) : string;
958: Function Format2(const Format:string;const Args:array of const;const FSettings:TFormatSettings): string
959: FUNCTION FormatInt(i: integer): STRING;
960: FUNCTION FormatInt64(i: int64): STRING;
961: Function FormatMaskText( const EditMask : string; const Value : string ) : string

```

```

962: Function FormatValue( AValue : Cardinal ) : string
963: Function FormatVersionString( const HiV, LoV : Word ) : string;
964: Function FormatVersionString1( const Major, Minor, Build, Revision : Word ) : string;
965: function Frac(X: Extended): Extended;
966: Function FreeResource( ResData : HGLOBAL ) : LongBool
967: Function FromCommon( const AValue : Double ) : Double
968: function FromCommon(const AValue: Double): Double;
969: Function FTPGMTDateTimeToMLS( const ATimeStamp : TDateTime; const AIncludeMSecs : Boolean ) : String
970: Function FTPLocalDateTimeToMLS( const ATimeStamp : TDateTime; const AIncludeMSecs : Boolean ) : String
971: Function FTPMLSToGMTDateTime( const ATimeStamp : String ) : TDateTime
972: Function FTPMLSToLocalDateTime( const ATimeStamp : String ) : TDateTime
973: Function FuncIn(AValue: Variant; ASet: Variant): Boolean;
974: //Function Funclist Size is: 6444 of mX3.9.8.9
975: Function FutureValue(const Rate:Extended;NPeriods:Integer;const Payment,PresentValue:Extended;PaymentTime:
TPaymentTime):Extended
976: Function Gauss( const x, Spread : Double ) : Double
977: function Gauss(const x,Spread: Double): Double;
978: Function GCD(x, y : LongInt ) : LongInt;
979: Function GCDJ( X, Y : Cardinal ) : Cardinal
980: Function GDAL: LongWord
981: Function GdiFlush : BOOL
982: Function GdiSetBatchLimit( Limit : DWORD ) : DWORD
983: Function GdiGetBatchLimit : DWORD
984: Function GenerateHeader : TIdHeaderList
985: Function GeometricMean( const X : TDynFloatArray ) : Float
986: Function Get( AURL : string ) : string;
987: Function Get2( AURL : string ) : string;
988: Function Get8087CW : Word
989: function GetActiveOleObject(const ClassName: String): IDispatch;
990: Function GetAliasDriverName( const AliasName : string ) : string
991: Function GetAPMBatteryFlag : TAPMBatteryFlag
992: Function GetAPMBatteryFullLifeTime : DWORD
993: Function GetAPMBatteryLifePercent : Integer
994: Function GetAPMBatteryLifeTime : DWORD
995: Function GetAPMLineStatus : TAPMLineStatus
996: Function GetAppdataFolder : string
997: Function GetAppDispatcher : TComponent
998: function GetArrayLength: integer;
999: Function GetASCII: string;
1000: Function GetASCIILine: string;
1001: Function GetAsHandle( Format : Word ) : THandle
1002: Function GetAssociatedProgram(const Extension: string; var Filename, Description: string): boolean;
1003: Function GetBackupFileName( const FileName : string ) : string
1004: Function GetBBitmap( Value : TBitmap ) : TBitmap
1005: Function GetBIOSCopyright : string
1006: Function GetBIOSDate : TDateTime
1007: Function GetBIOSExtendedInfo : string
1008: Function GetBIOSName : string
1009: Function getBitmap( apath: string ): TBitmap;
1010: Function GetBitmap( Index : Integer; Image : TBitmap ) : Boolean //object
1011: Function getBitmapObject(const bitmappath: string): TBitmap;
1012: Function GetButtonState( Button : TPageScrollerButton ) : TPageScrollerButtonState
1013: Function GetCapsLockKeyState : Boolean
1014: function GetCaptureControl: TControl;
1015: Function GetCDAudioTrackList( var TrackList : TJclCdTrackInfoArray; Drive : Char ) : TJclCdTrackInfo;
1016: Function GetCDAudioTrackList1( TrackList : TStrings; IncludeTrackType : Boolean; Drive : Char ) : string;
1017: Function GetCdInfo( InfoType : TJclCdMediaInfo; Drive : Char ) : string
1018: Function GetChangedText( const Text : string; SelStart, SelLength : Integer; Key : Char ) : string
1019: Function GetClientThread( ClientSocket : TServerClientWinSocket ) : TServerClientThread
1020: Function GetClockValue : Int64
1021: function getCmdLine: PChar;
1022: function getCmdShow: Integer;
1023: function GetCPUSpeed: Double;
1024: Function GetColField( DataCol : Integer ) : TField
1025: Function GetColorBlue( const Color : TColor ) : Byte
1026: Function GetColorFlag( const Color : TColor ) : Byte
1027: Function GetColorGreen( const Color : TColor ) : Byte
1028: Function GetColorRed( const Color : TColor ) : Byte
1029: Function GetComCtlVersion : Integer
1030: Function GetComPorts: TStringlist;
1031: Function GetCommonAppdataFolder : string
1032: Function GetCommonDesktopdirectoryFolder : string
1033: Function GetCommonFavoritesFolder : string
1034: Function GetCommonFilesFolder : string
1035: Function GetCommonProgramsFolder : string
1036: Function GetCommonStartmenuFolder : string
1037: Function GetCommonStartupFolder : string
1038: Function GetComponent( Owner, Parent : TComponent ) : TComponent
1039: Function GetConnectionRegistryFile( DesignMode : Boolean ) : string
1040: Function GetCookiesFolder : string
1041: Function GetCPUSpeed( var CpuSpeed : TFreqInfo ) : Boolean
1042: Function GetCurrent : TFavoriteLinkItem
1043: Function GetCurrent : TListItem
1044: Function GetCurrent : TTaskDialogBaseButtonItem
1045: Function GetCurrent : TToolButton
1046: Function GetCurrent : TTreeNode
1047: Function GetCurrent : WideString
1048: Function GetCurrentDir : string
1049: function GetCurrentDir: string)

```



```

1050: Function GetCurrentFolder : string
1051: Function GETCURRENTRECORD( BUFFER : PCHAR ) : BOOLEAN
1052: Function GetCurrentProcessId : TIdPID
1053: Function GetCurrentThreadHandle : THandle
1054: Function GetCurrentThreadId : LongWord; stdcall;
1055: Function GetCustomHeader( const Name : string ) : String
1056: Function GetDataItem( Value : Pointer ) : Longint
1057: Function GetDataLinkFiles( FileNames : TWideStrings; Directory : string ) : Integer;
1058: Function GetDataLinkFiles1( FileNames : TStrings; Directory : string ) : Integer;
1059: Function GETDATASIZE : INTEGER
1060: Function GetDC(hdwnd: HWND): HDC;
1061: Function GetDefaultFileExt( const MIMEType : string ) : string
1062: Function GetDefaults : Boolean
1063: Function GetDefaultSchemaName : WideString
1064: Function GetDefaultStreamLoader : IStreamLoader
1065: Function GetDesktopDirectoryFolder : string
1066: Function GetDesktopFolder : string
1067: Function GetDFASState( oStates : TList ) : TniRegularExpressionState
1068: Function GetDirectorySize( const Path : string ) : Int64
1069: Function GetDisplayWidth : Integer
1070: Function GetDLLVersion( const DLLName : string; var pdwMajor, pdwMinor : Integer ) : Boolean
1071: Function GetDomainName : string
1072: Function GetDriverRegistryFile( DesignMode : Boolean ) : string
1073: function GetDriveType(rootpath: pchar): cardinal;
1074: Function GetDriveTypeStr( const Drive : Char ) : string
1075: Function GetEnumerator : TFavoriteLinkItemsEnumerator
1076: Function GetEnumerator : TListItemsEnumerator
1077: Function GetEnumerator : TTaskDialogButtonsEnumerator
1078: Function GetEnumerator : TToolBarEnumerator
1079: Function GetEnumerator : TTreeNodeEnumerator
1080: Function GetEnumerator : TWideStringsEnumerator
1081: Function GetEnvVar( const VarName : string ) : string
1082: Function GetEnvironmentVar( const AVariableName : string ) : string
1083: Function GetEnvironmentVariable( const VarName : string ) : string
1084: Function GetEnvironmentVar( const Name : string; var Value : string; Expand : Boolean ) : Boolean
1085: Function GetEnvironmentVars( const Vars : TStrings; Expand : Boolean ) : Boolean
1086: Function getEnvironmentString: string;
1087: Function GetExceptionHandler : TObject
1088: Function GetFavoritesFolder : string
1089: Function GetFieldByName( const Name : string ) : string
1090: Function GetFieldInfo( const Origin : WideString; var FieldInfo : TFieldInfo ) : Boolean
1091: Function GetFieldValue( ACol : Integer ) : string
1092: Function GetFileAgeCoherence( const FileName : string ) : Boolean
1093: Function GetFileCreation( const FileName : string ) : TFileTime
1094: Function GetFileCreationTime( const Filename : string ) : TDateTime
1095: Function GetFileInformation( const FileName : string ) : TSearchRec
1096: Function GetFileLastAccess( const FileName : string ) : TFileTime
1097: Function GetFileLastWrite( const FileName : string ) : TFileTime
1098: Function GetFileList(FileList: TStringlist; apath: string): TStringlist;
1099: Function GetFileList1(apath: string): TStringlist;
1100: Function GetFileMIMEType( const AFileName : string ) : string
1101: Function GetFileSize( const FileName : string ) : Int64
1102: Function GetFileVersion( AFileName : string ) : Cardinal
1103: Function GetFileVersion( const AFilename : string ) : Cardinal
1104: Function GetFileSize2(Handle: Integer; x: Integer): Integer; stdcall;
1105: Function GetFileDate(aFile:string; aWithTime:Boolean):string;
1106: Function GetFilterData( Root : PExprNode ) : TExprData
1107: Function getFirstChild : LongInt
1108: Function getFirstChild : TTreeNode
1109: Function GetFirstDelimitedToken( const cDelim : char; const cStr : string ) : string
1110: Function GetFirstNode : TTreeNode
1111: Function GetFontsFolder : string
1112: Function GetFormulaValue( const Formula : string ) : Extended
1113: Function GetFreePageFileMemory : Integer
1114: Function GetFreePhysicalMemory : Integer
1115: Function GetFreeSystemResources( const ResourceType : TFreeSysResKind ) : Integer;
1116: Function GetFreeSystemResources1 : TFreeSystemResources;
1117: Function GetFreeVirtualMemory : Integer
1118: Function GetFromClipboard : Boolean
1119: Function GetFullURI( const AOptionalFields : TIdURIOptionalFieldsSet ) : String
1120: Function GetGBitmap( Value : TBitmap ) : TBitmap
1121: Function GetGMTDateByName( const AFileName : TIdFileName ) : TDateTime
1122: Function GetGroupState( Level : Integer ) : TGroupPosInds
1123: Function GetHandle : HWND
1124: Function GETHELPCONTEXT( VALUE : INTEGER; BYCOMMAND : BOOLEAN ) : THELPCONTEXT
1125: function GetHexArray(ahexdig: THexArray): THexArray;
1126: Function GetHighLightColor( const Color : TColor; Luminance : Integer ) : TColor
1127: function GetHINSTANCE: longword;
1128: Function GetHistoryFolder : string
1129: Function GetHitTestInfoAt( X, Y : Integer ) : THitTests
1130: function getHMODULE: longword;
1131: Function GetHostByName(const AComputerName: String): String;
1132: Function GetHostName : string
1133: Function getHostIP: string;
1134: Function GetHotSpot : TPoint
1135: Function GetHueBitmap( Value : TBitmap ) : TBitmap
1136: Function GetImageBitmap : HBITMAP
1137: Function GETIMAGELIST : TCUSTOMIMAGELIST
1138: Function GetIncome( const aNetto : Currency ) : Currency

```

```

1139: Function GetIncome( const aNetto : Extended) : Extended
1140: Function GetIncome( const aNetto : Extended): Extended
1141: Function GetIncome(const aNetto : Extended) : Extended
1142: function GetIncome(const aNetto: Currency): Currency
1143: Function GetIncome2( const aNetto : Currency) : Currency
1144: Function GetIncome2( const aNetto : Currency): Currency
1145: Function getIndex_Attrs( tag : string; var idx : Integer; var Attrs : string) : string
1146: Function GETINDEXFORFIELDS( const FIELDS : String; CASEINSENSITIVE : BOOLEAN) : TINDEXDEF
1147: Function GetIndexForOrderBy( const SQL : WideString; DataSet : TDataSet) : TIndexDef
1148: Function GetInstRes(Instance:THandle;ResType:TResType;const
    Name:string;Width:Integer;LoadFlags:TLoadResources;MaskColor:TColor):Boolean;
1149: Function
    GetInstRes1(Instance:THandle;ResType:TResType;ResID:DWORD;Width:Integer;LoadFlags:TLoadResources;MaskColor:
    TColor):Boolean;
1150: Function GetIntelCacheDescription( const D : Byte) : string
1151: Function GetInteractiveUserName : string
1152: Function GetInternetCacheFolder : string
1153: Function GetInternetFormattedFileTimeStamp( const AFilename : String) : String
1154: Function GetIPAddress( const HostName : string) : string
1155: Function GetIP( const HostName : string) : string
1156: Function GetIPHostByName(const AComputerName: String): String;
1157: Function GetIsAdmin: Boolean;
1158: Function GetItem( X, Y : Integer) : LongInt
1159: Function GetItemAt( X, Y : Integer) : TListItem
1160: Function GetItemHeight(Font: TFont): Integer;
1161: Function GetItemPath( Index : Integer) : string
1162: Function GetKeyFieldNames( List : TStringList) : Integer;
1163: Function GetKeyFieldNames1( List : TWideStrings) : Integer;
1164: Function GetKeyState( const VirtualKey : Cardinal) : Boolean
1165: Function GetLastChild : LongInt
1166: Function GetLastChild : TTreeNode
1167: Function GetLastDelimitedToken( const cDelim : char; const cStr : string) : string
1168: function GetLastError: Integer
1169: Function GetLAT_CONV_FACTOR: double; //for WGS84 power(1 - 1 / 298.257223563, 2);
1170: Function GetLoader( Ext : string) : TBitmapLoader
1171: Function GetLoadFilter : string
1172: Function GetLocalComputerName : string
1173: Function GetLocaleChar( Locale, LocaleType : Integer; Default : Char) : Char
1174: Function GetLocaleStr( Locale, LocaleType : Integer; Default : string) : string
1175: Function GetLocalUserName : string
1176: Function GetLoginUsername : WideString
1177: function getLongDayNames: string
1178: Function GetLongHint(const hint: string): string
1179: function getLongMonthNames: string
1180: Function GetMacAddresses( const Machine : string; const Addresses : TStringList) : Integer
1181: Function GetMainAppWndFromPid( PID : DWORD) : HWND
1182: Function GetMaskBitmap : HBITMAP
1183: Function GetMaxAppAddress : Integer
1184: Function GetMciErrorMessage( const MciErrNo : MCIERROR) : string
1185: Function GetMemoryLoad : Byte
1186: Function GetMIMEDefaultFileExt( const MIMETYPE : string) : TIdFileName
1187: Function GetMIMETYPEFromFile( const AFile : string) : string
1188: Function GetMIMETYPEFromFile( const AFile : TIdFileName) : string
1189: Function GetMinAppAddress : Integer
1190: Function GetModule : TComponent
1191: Function GetModuleHandle( ModuleName : PChar) : HMODULE
1192: Function GetModuleName( Module : HMODULE) : string
1193: Function GetModulePath( const Module : HMODULE) : string
1194: Function GetModuleFileName(Module: Integer; Filename: PChar;Size: Integer): Integer; stdcall;
1195: Function GetCommandLine: PChar; stdcall;
1196: Function GetMonochromeBitmap( Value : TBitmap) : TBitmap
1197: Function GetMultiN(aval: integer): string;
1198: Function GetName : String
1199: Function GetNearestItem( Point : TPoint; Direction : TSearchDirection) : TListItem
1200: Function GetNethoodFolder : string
1201: Function GetNext : TTreeNode
1202: Function GetNextChild( Value : LongInt) : LongInt
1203: Function GetNextChild( Value : TTreeNode) : TTreeNode
1204: Function GetNextDelimitedToken( const cDelim : char; var cStr : String) : String
1205: Function GetNextItem( StartItem: TListItem;Direction:TSearchDirection;States:TItemStates) : TListItem
1206: Function GetNextPacket : Integer
1207: Function getNextSibling : TTreeNode
1208: Function GetNextVisible : TTreeNode
1209: Function GetNode( ItemId : HTreeItem) : TTreeNode
1210: Function GetNodeAt( X, Y : Integer) : TTreeNode
1211: Function GetNodeDisplayWidth( Node : TOutlineNode) : Integer
1212: function GetNumberOfProcessors: longint;
1213: Function GetNumLockKeyState : Boolean
1214: Function GetObjectProperty( Instance : TPersistent; const PropName : string) : TObject
1215: Function GetOnlyTransitionOn( cChar : char) : TniRegularExpressionState
1216: Function GetOptionalParam( const ParamName : string) : OleVariant
1217: Function GetOSName: string;
1218: Function GetOSVersion: string;
1219: Function GetOSNumber: string;
1220: Function GetOsVersionInfo: TOSVersionInfo; //thx to wischniewski
1221: Function GetPackageModuleHandle( PackageName : PChar) : HMODULE
1222: function GetPageSize: Cardinal;
1223: Function GetParameterFileName : string
1224: Function GetParams( var OwnerData : OleVariant) : OleVariant

```

```

1225: Function GETPARENTCOMPONENT : TCOMPONENT
1226: Function GetParentForm(control: TControl): TForm
1227: Function GETPARENTMENU : TMENU
1228: Function GetPassword : Boolean
1229: Function GetPassword : string
1230: Function GetPersonalFolder : string
1231: Function GetPidFromProcessName( const ProcessName : string) : DWORD
1232: Function GetPosition : TPoint
1233: Function GetPrev : TTreeNode
1234: Function GetPrevChild( Value : LongInt) : LongInt
1235: Function GetPrevChild( Value : TTreeNode) : TTreeNode
1236: Function getPrevSibling : TTreeNode
1237: Function GetPrevVisible : TTreeNode
1238: Function GetPrinthoodFolder : string
1239: Function GetPrivilegeDisplayName( const PrivilegeName : string) : string
1240: Function getProcessList: TStrings;
1241: Function GetProcessId : TidPID
1242: Function GetProcessNameFromPid( PID : DWORD) : string
1243: Function GetProcessNameFromWnd( Wnd : HWND) : string
1244: Function GetProcessMemoryInfo(Process: THandle;ppsmemCounters: TProcessMemoryCounters;cb: DWORD):BOOL
1245: Function getProcessAllMemory(ProcessID : DWORD): TProcessMemoryCounters;
1246: Function getProcessMemoryInfo2(ProcessID : DWORD): TProcessMemoryCounters;
1247: Function GetProgramFilesFolder : string
1248: Function GetProgramsFolder : string
1249: Function GetProxy : string
1250: Function GetQuoteChar : WideString
1251: Function GetQrCode4(Width,Height:Word; Correct_Level:string;const
Data:string;aformat:string):TLinearBitmap;
1252: Function GetQrCodeToFile(Width,Height:Word;Correct_Level:string;const
Data:string;aformat:string):TLinearBitmap;
1253: Function GetRate : Double
1254: Function getPerfTime: string;
1255: Function getRuntime: string;
1256: Function GetRBitmap( Value : TBitmap) : TBitmap
1257: Function GetReadableName( const AName : string) : string
1258: Function GetRecentDocs : TStringList
1259: Function GetRecentFolder : string
1260: Function GetRecords( Count : Integer; out RecsOut : Integer; Options : Integer) : OleVariant;
1261: Function GetRecords1(Count:Integer; out RecsOut:Integer;Options:Integer;const CommandText:WideString;var
Params, OwnerData : OleVariant) : OleVariant;
1262: Function GetRecordset( const CommandText : WideString; ConnectionString : WideString) : _Recordset
1263: Function GetRegisteredCompany : string
1264: Function GetRegisteredOwner : string
1265: Function GetResource(ResType:TResType;const
Name:string;Width:Integer;LoadFlags:TLoadResources;MaskColor:TColor:Boolean
1266: Function GetResourceName( ObjStream : TStream; var AName : string) : Boolean
1267: Function GetResponse( const AAllowedResponses : array of SmallInt) : SmallInt;
1268: Function GetResponse1( const AAllowedResponse : SmallInt) : SmallInt;
1269: Function GetRValue( rgb : DWORD) : Byte
1270: Function GetGValue( rgb : DWORD) : Byte
1271: Function GetBValue( rgb : DWORD) : Byte
1272: Function GetCValue( cmyk : COLORREF) : Byte
1273: Function GetMValue( cmyk : COLORREF) : Byte
1274: Function GetYValue( cmyk : COLORREF) : Byte
1275: Function GetKValue( cmyk : COLORREF) : Byte
1276: Function CMYK( c, m, y, k : Byte) : COLORREF
1277: Function GetOSName: string;
1278: Function GetProcAddress( hModule : HMODULE; lpProcName : LPCSTR) : FARPROC
1279: Function GetProcAddress(Module : HMODULE; Proc : PChar): Dword
1280: Function GetSafeCallExceptionMsg : String
1281: Function GetSaturationBitmap( Value : TBitmap) : TBitmap
1282: Function GetSaveFilter : string
1283: Function GetSaver( Ext : string) : TBitmapLoader
1284: Function GetScrollLockKeyState : Boolean
1285: Function GetSearchString : string
1286: Function GetSelections( AList : TList) : TTreeNode
1287: Function GETSELTEXTBUF(BUFFER:PCHAR;BUFSIZE:INTEGER):INTEGER
1288: Function GetSendToFolder : string
1289: Function GetServer : IAppServer
1290: Function GetServerList : OleVariant
1291: Function GetShadowColor( const Color : TColor; Luminance : Integer) : TColor
1292: Function GetShellProcessHandle : THandle
1293: Function GetShellProcessName : string
1294: Function GetShellVersion : Cardinal
1295: function getShortDayNames: string
1296: Function GetShortHint(const hint: string): string
1297: function getShortMonthNames: string
1298: Function GetSizeOfFile( const FileName : string) : Int64;
1299: Function GetSizeOfFile1( Handle : THandle) : Int64;
1300: Function GetStdHandle(nStdHandle: Integer): Integer; stdcall;
1301: Function GetStartmenuFolder : string
1302: Function GetStartupFolder : string
1303: Function GetStringProperty( Instance : TPersistent; const PropName : string) : WideString
1304: Function GetSuccessor( cChar : char) : TniRegularExpressionState
1305: Function GetSwapFileSize : Integer
1306: Function GetSwapFileUsage : Integer
1307: Function GetSystemLocale : TidCharSet
1308: Function GetSystemMetrics( nIndex : Integer) : Integer
1309: Function GetSystemPathSH(Folder: Integer): TFilename ;

```

```

1310: Function GetTableNameFromQuery( const SQL : WideString) : WideString
1311: Function GetTableNameFromSQL( const SQL : WideString) : WideString
1312: Function GetTableNameFromSQLEx( const SQL : WideString; IdOption : IDENTIFIEROption) : WideString
1313: Function GetTasksList( const List : TStringList) : Boolean
1314: Function getTeamViewerID: string;
1315: Function GetTemplatesFolder : string
1316: Function GetText : PWideChar
1317: function GetText:PChar
1318: Function GetTextBuf( Buffer : PChar; BufSize : Integer) : Integer
1319: function GETTEXTBUF(BUFFER:PCHAR;BUFSIZE:INTEGER):INTEGER
1320: Function GetTextItem( const Value : string) : Longint
1321: function GETTEXTLEN:INTEGER
1322: Function GetThreadLocale: Longint; stdcall
1323: Function GetCurrentThreadID: LongWord; stdcall;
1324: Function GetTickCount : Cardinal
1325: Function GetTickCountDiff( const AOldTickCount, ANewTickCount : Cardinal) : Cardinal
1326: Function GetTicketNr : longint
1327: Function GetTime : Cardinal
1328: Function GetTime : TDateTime
1329: Function GetTimeout : Integer
1330: Function GetTimeStr: String
1331: Function GetTimeString: String
1332: Function GetTodayFiles(startdir, amask: string): TStringlist;
1333: Function getTokenCounts : integer
1334: Function GetTotalPageFileMemory : Integer
1335: Function GetTotalPhysicalMemory : Integer
1336: Function GetTotalVirtualMemory : Integer
1337: Function GetUniqueFileName( const APath, APrefix, AExt : String) : String
1338: Function GetUseNowForDate : Boolean
1339: Function GetUserDomainName( const CurUser : string) : string
1340: Function GetUserName : string
1341: Function GetUserName: string;
1342: Function GetUserObjectName( hUserObject : THandle) : string
1343: Function GetValueBitmap( Value : TBitmap) : TBitmap
1344: Function GetValueMSec : Cardinal
1345: Function GetValueStr : String
1346: Function GetVersion: int;
1347: Function GetVersionString(FileName: string): string;
1348: Function GetVisibleNode( Index : LongInt) : TOutlineNode
1349: Function GetVolumeFileSystem( const Drive : string) : string
1350: Function GetVolumeName( const Drive : string) : string
1351: Function GetVolumeSerialNumber( const Drive : string) : string
1352: Function GetWebAppServices : IWebAppServices
1353: Function GetWebRequestHandler : IWebRequestHandler
1354: Function GetWindowCaption( Wnd : HWND) : string
1355: Function GetWindowDC(hwnd: HWND): HDC;
1356: Function GetWindowIcon( Wnd : HWND; LargeIcon : Boolean) : HICON
1357: Function GetWindowRect(hwnd: HWND; arect: TRect): Boolean
1358: Function GetWindowsComputerID : string
1359: function GetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
1360: Function GetWindowsFolder : string
1361: Function GetWindowsServicePackVersion : Integer
1362: Function GetWindowsServicePackVersionString : string
1363: Function GetWindowsSystemFolder : string
1364: Function GetWindowsTempFolder : string
1365: Function GetWindowsUserID : string
1366: Function GetWindowsVersion : TWindowsVersion
1367: Function GetWindowsVersionString : string
1368: Function GmtOffsetStrToDateTime( S : string) : TDateTime
1369: Function GMTToLocalDateTime( S : string) : TDateTime
1370: Function GotoKey : Boolean
1371: Function GradToCycle( const Grads : Extended) : Extended
1372: Function GradToDeg( const Grads : Extended) : Extended
1373: Function GradToDeg( const Value : Extended) : Extended;
1374: Function GradToDeg1( const Value : Double) : Double;
1375: Function GradToDeg2( const Value : Single) : Single;
1376: Function GradToRad( const Grads : Extended) : Extended
1377: Function GradToRad( const Value : Extended) : Extended;
1378: Function GradToRad1( const Value : Double) : Double;
1379: Function GradToRad2( const Value : Single) : Single;
1380: Function Gray32( const Intensity : Byte; const Alpha : Byte) : TColor32
1381: Function GreenComponent( const Color32 : TColor32) : Integer
1382: Function GUIDToString(const GUID: TGUID): string)
1383: Function HandleAllocated : Boolean
1384: function HandleAllocated: Boolean;
1385: Function HandleRequest : Boolean
1386: Function HandleRequest( Request : TWebRequest; Response : TWebResponse) : Boolean
1387: Function HarmonicMean( const X : TDynFloatArray) : Float
1388: Function HasAsParent( Value : TTreeNode) : Boolean
1389: Function HASCHILDEFS : BOOLEAN
1390: Function HasCurValues : Boolean
1391: Function HasExtendCharacter( const s : UTF8String) : Boolean
1392: Function HasFormat( Format : Word) : Boolean
1393: Function HashValue( AStream : TStream) : T5x4LongWordRecord;
1394: Function HashValue(AStream : TStream) : T4x4LongWordRecord
1395: Function HashValue(AStream: TStream): LongWord
1396: Function HashValue(AStream: TStream): Word
1397: Function HashValue1( AStream : TStream; const ABeginPos, AEndPos : Int64) : T5x4LongWordRecord;
1398: Function HashValue1(AStream : TStream) : T4x4LongWordRecord

```



```

1399: Function HashValue128(const ASrc: string): T4x4LongWordRecord;
1400: Function HashValue128Stream(AStream: TStream): T4x4LongWordRecord;
1401: Function HashValue16(const ASrc: string): Word;
1402: Function HashValue16Stream(AStream: TStream): Word;
1403: Function HashValue32(const ASrc: string): LongWord;
1404: Function HashValue32Stream(AStream: TStream): LongWord;
1405: Function HasMergeConflicts: Boolean
1406: Function hasMoreTokens: boolean
1407: Function HASPARENT: BOOLEAN
1408: Function HasParent: Boolean
1409: Function HasTransaction( Transaction: TDBXTransaction): Boolean
1410: Function HasUTF8BOM( S: TStream): boolean;
1411: Function HasUTF8BOM1( S: AnsiString): boolean;
1412: Function Haversine( X: Float): Float
1413: Function Head( s: string; const subs: string; var tail: string): string
1414: Function HELPCOMMAND(COMMAND:INTEGER;DATA:LONGINT):BOOLEAN
1415: Function HELPCONTEXT(CONTEXT:THELPCONTEXT):BOOLEAN
1416: Function HELPJUMP(JUMPID:STRING):BOOLEAN
1417: Function HeronianMean(const a, b: Float): Float
1418: Function HexStrToStr(Value: string): string;
1419: Function HexToBin(Text, Buffer: PChar; BufSize: Integer): Integer;
1420: Function HexToBin2(HexNum: string): string;
1421: Function HexToDouble(const Hex: string): Double
1422: Function HexToInt(hexnum: string): LongInt;
1423: Function HexToStr(Value: string): string;
1424: Function HexifyBlock(var Buffer, BufferSize: Integer): string
1425: Function Hi(vdat: word): byte;
1426: Function HiByte(W: Word): Byte)
1427: Function High: Int64;
1428: Function HighlightCell(DataCol, DataRow: Integer; const Value: string; AState: TGridDrawState): Boolean
1429: Function HINSTANCE: longword;
1430: Function HiWord(l: DWORD): Word)
1431: Function HMODULE: longword;
1432: Function HourOf(const AValue: TDateTime): Word
1433: Function HourOfDay(const AValue: TDateTime): Word
1434: Function HourOfTheMonth(const AValue: TDateTime): Word
1435: Function HourOfTheWeek(const AValue: TDateTime): Word
1436: Function HourOfTheYear(const AValue: TDateTime): Word
1437: Function HoursBetween(const ANow, AThen: TDateTime): Int64
1438: Function HourSpan(const ANow, AThen: TDateTime): Double
1439: Function HSLToRGB1(const H, S, L: Single): TColor32;
1440: Function HTMLDecode(const AStr: String): String
1441: Function HTMLEncode(const AStr: String): String
1442: Function HTMLEscape(const Str: string): string
1443: Function HtmlTable(DataSet: TDataSet; DataSetHandler: TDSTableProducer; MaxRows: Integer): string
1444: Function HTTPDecode(const AStr: String): string
1445: Function HTTPEncode(const AStr: String): string
1446: Function Hypot(const X, Y: Extended): Extended
1447: Function IBMax(n1, n2: Integer): Integer
1448: Function IBMin(n1, n2: Integer): Integer
1449: Function IBRandomString(iLength: Integer): String
1450: Function IBRandomInteger(iLow, iHigh: Integer): Integer
1451: Function IBStripString(st: String; CharsToStrip: String): String
1452: Function IBFormatIdentifier(Dialect: Integer; Value: String): String
1453: Function IBFormatIdentifierValue(Dialect: Integer; Value: String): String
1454: Function IBExtractIdentifier(Dialect: Integer; Value: String): String
1455: Function IBQuoteIdentifier(Dialect: Integer; Value: String): String
1456: Function IBAddIBParamSQLForDetail(Params: TParams; SQL: string; Native: Boolean; Dialect: Integer): string
1457: Procedure IBDecomposeDatabaseName(DatabaseName: String; var ServerName, Protocol, DatabasePath: String)
1458: Function RandomString(iLength: Integer): String;
1459: Function RandomInteger(iLow, iHigh: Integer): Integer;
1460: Function StripString(st: String; CharsToStrip: String): String;
1461: Function FormatIdentifier(Dialect: Integer; Value: String): String;
1462: Function FormatIdentifierValue(Dialect: Integer; Value: String): String;
1463: Function ExtractIdentifier(Dialect: Integer; Value: String): String;
1464: Function QuoteIdentifier(Dialect: Integer; Value: String): String;
1465: Function AddIBParamSQLForDetail(Params: TParams; SQL: string; Native: Boolean; Dialect: Integer): string;
1466: Procedure DecomposeDatabaseName(DatabaseName: String; var ServerName, Protocol, DatabasePath: String);
1467: Function NextSQLToken(var p: PChar; var Token: String; CurSection: TSQLToken): TSQLToken;
1468: Function IconToBitmap(Ico: HICON): TBitmap
1469: Function IconToBitmap2(Ico: HICON; Size: Integer; TransparentColor: TColor): TBitmap
1470: Function IconToBitmap3(Ico: HICON; Size: Integer; TransparentColor: TColor): TBitmap
1471: Function IdentToCharset(const Ident: string; var CharSet: Longint): Boolean
1472: Function IdentToColor(const Ident: string; var Color: Longint): Boolean
1473: Function IdentToCursor(const Ident: string; var cursor: Longint): Boolean;
1474: Function IdGetDefaultCharSet: TIdCharSet
1475: Function IDispatchInvoke(Self: IDispatch; ProperSet: Boolean; const Name: String; Par: array of variant): variant
1476: Function IdPorts2: TStringList
1477: Function IdToMib(const Id: string): string
1478: Function IdSHA1Hash(aphash: string): string;
1479: Function IdHashSHA1(aphash: string): string;
1480: Function IfStr(const bCondition: boolean; const sTrue: string; const sFalse: string): string
1481: Function IfThen(AValue: Boolean; const ATrue: string; AFalse: string): string;
1482: Function iif1(ATest: Boolean; const ATrue: Integer; const AFalse: Integer): Integer;
1483: Function iif2(ATest: Boolean; const ATrue: string; const AFalse: string): string;
1484: Function iif3(ATest: Boolean; const ATrue: Boolean; const AFalse: Boolean): Boolean;
1485: Function ImportTest(S1: string; s2: longint; s3: Byte; s4: word; var s5: string): string;

```

```

1486: Function IncDay( const AValue : TDateTime; const ANumberOfDays : Integer ) : TDateTime
1487: Function IncHour( const AValue : TDateTime; const ANumberOfHours : Int64 ) : TDateTime
1488: Function IncLimit( var B : Byte; const Limit : Byte; const Incr : Byte ) : Byte;
1489: Function IncLimit1( var B : Shortint; const Limit : Shortint; const Incr : Shortint ) : Shortint;
1490: Function IncLimit2( var B : Smallint; const Limit : Smallint; const Incr : Smallint ) : Smallint;
1491: Function IncLimit3( var B : Word; const Limit : Word; const Incr : Word ) : Word;
1492: Function IncLimit4( var B : Integer; const Limit : Integer; const Incr : Integer ) : Integer;
1493: Function IncLimit5( var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal ) : Cardinal;
1494: Function IncLimit6( var B : Int64; const Limit : Int64; const Incr : Int64 ) : Int64;
1495: Function IncLimitClamp( var B : Byte; const Limit : Byte; const Incr : Byte ) : Byte;
1496: Function IncLimitClamp1( var B : Shortint; const Limit : Shortint; const Incr : Shortint ) : Shortint;
1497: Function IncLimitClamp2( var B : Smallint; const Limit : Smallint; const Incr : Smallint ) : Smallint;
1498: Function IncLimitClamp3( var B : Word; const Limit : Word; const Incr : Word ) : Word;
1499: Function IncLimitClamp4( var B : Integer; const Limit : Integer; const Incr : Integer ) : Integer;
1500: Function IncLimitClamp5( var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal ) : Cardinal;
1501: Function IncLimitClamp6( var B : Int64; const Limit : Int64; const Incr : Int64 ) : Int64;
1502: Function IncludeTrailingBackslash( S : string ) : string
1503: Function IncludeTrailingBackslash(const S : string) : string
1504: Function IncludeTrailingPathDelimiter( const APath : string ) : string
1505: Function IncludeTrailingPathDelimiter( S : string ) : string
1506: Function IncludeTrailingPathDelimiter(const S : string) : string
1507: Function IncludeTrailingSlash( const APath : string ) : string
1508: Function IncMilliSecond( const AValue : TDateTime; const ANumberOfMilliseconds : Int64 ) : TDateTime
1509: Function IncMinute( const AValue : TDateTime; const ANumberOfMinutes : Int64 ) : TDateTime
1510: Function IncMonth( DateTime : TDateTime; NumberOfMonths : Integer ) : TDateTime
1511: Function IncMonth(const DateTime: TDateTime; NumberOfMonths: Integer): TDateTime)
1512: Function IncSecond( const AValue : TDateTime; const ANumberOfSeconds : Int64 ) : TDateTime
1513: Function IncWeek( const AValue : TDateTime; const ANumberOfWeeks : Integer ) : TDateTime
1514: Function IncYear( const AValue : TDateTime; const ANumberOfYears : Integer ) : TDateTime
1515: Function IndexOf( AClass : TClass ) : Integer
1516: Function IndexOf( AComponent : TComponent ) : Integer
1517: Function IndexOf( AObject : TObject ) : Integer
1518: Function INDEXOF( const ANAME : String ) : INTEGER
1519: Function IndexOf( const DisplayName : string ) : Integer
1520: Function IndexOf( const Item : TBookmarkStr ) : Integer
1521: Function IndexOf( const S : WideString ) : Integer
1522: Function IndexOf( const View : TJclFileMapView ) : Integer
1523: Function INDEXOF( FIELD : TFIELD ) : INTEGER
1524: Function IndexOf( ID : LCID ) : Integer
1525: Function INDEXOF( ITEM : TMENUITEM ) : INTEGER
1526: Function IndexOf( Value : TListItem ) : Integer
1527: Function IndexOf( Value : TTreeNode ) : Integer
1528: Function IndexOf(const S : string) : Integer;
1529: Function IndexOfName( const Name : WideString ) : Integer
1530: Function IndexOfName(Name: string): Integer;
1531: Function IndexOfObject( AObject : TObject ) : Integer
1532: Function IndexofObject(AObject:TObject):Integer
1533: Function IndexOfTabAt( X, Y : Integer ) : Integer
1534: Function IndexStr( const AText : string; const AValues : array of string ) : Integer
1535: Function IndexText( const AText : string; const AValues : array of string ) : Integer
1536: Function IndexOfInteger( AList : TStringList; Value : Variant ) : Integer
1537: Function IndexOfFloat( AList : TStringList; Value : Variant ) : Integer
1538: Function IndexOfDate( AList : TStringList; Value : Variant ) : Integer
1539: Function IndexOfString( AList : TStringList; Value : Variant ) : Integer
1540: Function IndyCompareStr( const A1 : string; const A2 : string ) : Integer
1541: Function IndyGetHostName : string
1542: Function IndyInterlockedDecrement( var I : Integer ) : Integer
1543: Function IndyInterlockedExchange( var A : Integer; B : Integer ) : Integer
1544: Function IndyInterlockedExchangeAdd( var A : Integer; B : Integer ) : Integer
1545: Function IndyInterlockedIncrement( var I : Integer ) : Integer
1546: Function IndyLowerCase( const A1 : string ) : string
1547: Function IndyStrToBool( const AString : String ) : Boolean
1548: Function IndyUpperCase( const A1 : string ) : string
1549: Function InitCommonControl( CC : Integer ) : Boolean
1550: Function InitTempPath : string
1551: Function InMainThread : boolean
1552: Function InOpArray( W : WideChar; sets : array of WideChar ) : boolean
1553: Function Input : Text
1554: Function InputBox( const ACaption, APrompt, ADefault : string ) : string
1555: Function InputBox(const ACaption: string; const APrompt: string; const ADefault: string): string)
1556: Function InputLn(const AMask: string; AEcho:Boolean;ATabWidth:Integer;AMaxLineLength:Integer): string
1557: Function InputQuery( const ACaption, APrompt : string; var Value : string ) : Boolean
1558: Function InputQuery(const ACaption: string; const APrompt: string; var Value: string): Boolean)
1559: Function InquireSignal( RtlSigNum : Integer ) : TSignalState
1560: Function InRangeR( const A, Min, Max : Double ) : Boolean
1561: Function Insert( Index : Integer ) : TCollectionItem
1562: Function Insert( Index : Integer ) : TComboExItem
1563: Function Insert( Index : Integer ) : THeaderSection
1564: Function Insert( Index : Integer ) : TListItem
1565: Function Insert( Index : Integer ) : TStatusPanel
1566: Function Insert( Index : Integer ) : TWorkArea
1567: Function Insert( Index : LongInt; const Text : string ) : LongInt
1568: Function Insert( Sibling : TTreeNode; const S : string ) : TTreeNode
1569: Function INSERTNEWLINEAFTER( AITEM : TMENUITEM ) : INTEGER
1570: Function INSERTNEWLINEBEFORE( AITEM : TMENUITEM ) : INTEGER
1571: Function InsertNode( Node, Sibling : TTreeNode; const S : string; Ptr : Pointer ) : TTreeNode
1572: Function InsertObject( Index : LongInt; const Text : string; const Data : Pointer ) : LongInt
1573: Function InsertObject( Sibling : TTreeNode; const S : string; Ptr : Pointer ) : TTreeNode
1574: Function Instance : Longint

```

```

1575: function InstanceSize: Longint
1576: function Int(e : Extended) : Extended;
1577: function Int64ToStr(i: Int64): String;
1578: function IntegerToBcd( const AValue : Integer) : TBcd
1579: function Intensity( const Color32 : TColor32) : Integer;
1580: function Intensity( const R, G, B : Single) : Single;
1581: function InterestPayment(const Rate:Extended;Period,NPeriods:Integer;const PresentValue,
FutureValue:Extended; PaymentTime : TPaymentTime) : Extended
1582: function InterestRate(NPeriods:Integer;const Payment,PresentVal,
FutureVal:Extended;PaymentTime:TPaymentTime): Extended
1583: function InternalDecodeDate( DateTime : TDateTime; var Year, Month, Day, DOW : Word) : Boolean
1584: function InternalRateOfReturn( const Guess : Extended; const CashFlows : array of Double) : Extended
1585: function InternalUpdateRecord( Tree : TUpdateTree) : Boolean
1586: function IntersectRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
1587: function IntersectRect(out Rect: TRect; const R1, R2: TRect): Boolean)
1588: function IntMibToStr( const Value : string) : string
1589: function IntPower( const Base : Extended; const Exponent : Integer) : Extended
1590: function IntToBin( Value : cardinal) : string
1591: function IntToHex( Value : Integer; Digits : Integer) : string;
1592: function IntToHex(a: integer; b: integer): string;
1593: function IntToHex64( Value : Int64; Digits : Integer) : string;
1594: function IntToHex64(Value: Int64; Digits: Integer): string)
1595: function IntTo3Str( Value : Longint; separator: string) : string
1596: function inttobool( aInt : LongInt) : Boolean
1597: function IntToStr(i: Int64): String;
1598: function IntToStr64(Value: Int64): string)
1599: function IOResult: Integer
1600: function IPv6AddressToStr(const AValue: TIdIPv6Address): string
1601: function IsAccel(VK: Word; const Str: string): Boolean
1602: function IsAddressInNetwork( Address : string) : Boolean
1603: function IsAdministrator : Boolean
1604: function IsAlias( const Name : string) : Boolean
1605: function IsApplicationRunning( const AClassName, ApplName : string) : Boolean
1606: function IsASCII( const AByte : Byte) : Boolean;
1607: function IsASCIILDH( const AByte : Byte) : Boolean;
1608: function IsAssembly(const FileName: string): Boolean;
1609: function IsBcdNegative( const Bcd : TBcd) : Boolean
1610: function IsBinary(const AChar : Char) : Boolean
1611: function IsConsole: Boolean)
1612: function IsDelimiter( Delimiters, S : string; Index : Integer) : Boolean
1613: function IsDelimiter(const Delimiters: string; const S: string; Index: Integer): Boolean)
1614: function IsDelphiDesignMode : boolean
1615: function IsDelphiRunning : boolean
1616: function IsDFSState : boolean
1617: function IsDirectory( const FileName : string) : Boolean
1618: function IsDomain( const S : String) : Boolean
1619: function IsDragObject(Sender: TObject): Boolean;
1620: function IsEditing : Boolean
1621: function ISEMPTY : BOOLEAN
1622: function IsEqual( Value : TParameters) : Boolean
1623: function ISEQUAL( VALUE : TPARAMS) : BOOLEAN
1624: function IsEqualGUID(const guid1, guid2: TGUID): Boolean)
1625: function IsFirstNode : Boolean
1626: function IsFloatZero( const X : Float) : Boolean
1627: function IsFormatRegistered( Extension, AppID : string) : Boolean
1628: function IsFormOpen(const FormName: string): Boolean;
1629: function IsFQDN( const S : String) : Boolean
1630: function IsGrayScale : Boolean
1631: function IsHex( AChar : Char) : Boolean;
1632: function IsHexString(const AString: string): Boolean;
1633: function IsHostname( const S : String) : Boolean
1634: function IsInfinite( const AValue : Double) : Boolean
1635: function IsInLeapYear( const AValue : TDateTime) : Boolean
1636: function IsInternet: boolean;
1637: function IsLeadChar( ACh : Char) : Boolean
1638: function IsLeapYear( Year : Word) : Boolean
1639: function IsLeapYear(Year: Word): Boolean)
1640: function IsLibrary: Boolean)
1641: function ISLINE : BOOLEAN
1642: function IsLinkedTo( DataSet : TDataSet) : Boolean
1643: function ISLINKEDTO( DATASOURCE : TDATASOURCE) : BOOLEAN
1644: function IsLiteralChar( const EditMask : string; Offset : Integer) : Boolean
1645: function IsMatch( const Pattern, Text : string) : Boolean //Grep like RegEx
1646: function IsMainAppWindow( Wnd : HWND) : Boolean
1647: function IsMediaPresentInDrive( Drive : Char) : Boolean
1648: function IsMemoryManagerSet: Boolean)
1649: function IsMultiTableQuery( const SQL : WideString) : Boolean
1650: function IsMultiThread: Boolean)
1651: function IsNumeric( AChar : Char) : Boolean;
1652: function IsNumeric2( const AString : string) : Boolean;
1653: function IsOctal( AChar : Char) : Boolean;
1654: function IsOctalString(const AString: string) : Boolean;
1655: function IsPathDelimiter( S : string; Index : Integer) : Boolean
1656: function IsPathDelimiter(const S: string; Index: Integer): Boolean)
1657: function IsPM( const AValue : TDateTime) : Boolean
1658: function IsPositiveFloatArray( const X : TDynFloatArray) : Boolean
1659: function IsPrimeFactor( const F, N : Cardinal) : Boolean
1660: function IsPrimeRM( N : Cardinal) : Boolean //rabin miller
1661: function IsPrimeTD( N : Cardinal) : Boolean //trial division

```

```

1662: Function IsPrivilegeEnabled( const Privilege : string ) : Boolean
1663: Function ISqrt( const I : Smallint ) : Smallint
1664: Function IsReadOnly(const Filename: string): boolean;
1665: Function IsRectEmpty( const Rect : TRect ) : Boolean
1666: Function IsRectEmpty(const Rect: TRect): Boolean
1667: Function IsRelativePrime( const X, Y : Cardinal ) : Boolean
1668: Function ISRIGHTTOLEFT : BOOLEAN
1669: Function IsRightToLeft: Boolean
1670: Function IsSameDay( const AValue, ABasis : TDateTime ) : Boolean
1671: Function ISSEQUENCED : BOOLEAN
1672: Function IsSystemModule( const Module : HMODULE ) : Boolean
1673: Function IsSystemResourcesMeterPresent : Boolean
1674: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
1675: Function IsToday( const AValue : TDateTime ) : Boolean
1676: Function IsToday(const AValue: TDateTime): Boolean;
1677: Function IsTopDomain( const AStr : string ) : Boolean
1678: Function IsUTF8LeadByte( Lead : Char ) : Boolean
1679: Function IsUTF8String( const s : UTF8String ) : Boolean
1680: Function IsUTF8TrailByte( Lead : Char ) : Boolean
1681: Function ISVALIDCHAR( INPUTCHAR : CHAR ) : BOOLEAN
1682: Function IsValidDate( const AYear, AMonth, ADay : Word ) : Boolean
1683: Function IsValidDateDay( const AYear, ADayOfYear : Word ) : Boolean
1684: Function IsValidDateMonthWeek( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word ) : Boolean
1685: Function IsValidDateTime(const AYear, AMonth, ADay, AHour, AMinute, ASecond, AMilliSecond: Word): Boolean
1686: Function IsValidDateWeek( const AYear, AWeekOfYear, ADayOfWeek : Word ) : Boolean
1687: Function IsValidIdent( Ident : string ) : Boolean
1688: function IsValidIdent(const Ident: string; AllowDots: Boolean): Boolean
1689: Function IsValidIP( const S : String ) : Boolean
1690: Function IsValidTime( const AHour, AMinute, ASecond, AMilliSecond : Word ) : Boolean
1691: Function IsValidISBN( const ISBN : AnsiString ) : Boolean
1692: Function IsVariantManagerSet: Boolean; //deprecated;
1693: Function IsVirtualPcGuest : Boolean;
1694: Function IsVmWareGuest : Boolean;
1695: Function IsVCLControl(Handle: HWND): Boolean;
1696: Function IsWhiteString( const AStr : String ) : Boolean
1697: Function IsWindowResponding( Wnd : HWND; Timeout : Integer ) : Boolean
1698: Function IsWoW64: boolean;
1699: Function IsWin64: boolean;
1700: Function IsWow64String(var s: string): Boolean;
1701: Function IsWin64String(var s: string): Boolean;
1702: Function IsWindowsVista: boolean;
1703: Function isPowerof2(num: int64): boolean;
1704: Function powerOf2(exponent: integer): int64;
1705: function IsZero(const A: Extended; Epsilon: Extended): Boolean //overload;
1706: function IsZero1(const A: Double; Epsilon: Double): Boolean //overload;
1707: function IsZero2(const A: Single; Epsilon: Single): Boolean //overload;
1708: Function ItemAtPos( Pos : TPoint; IgnoreTabHeight : Boolean ) : Integer
1709: function ITEMATPOS(POS:TPOINT;EXISTING:BOOLEAN):INTEGER
1710: Function ItemRect( Index : Integer ) : TRect
1711: function ITEMRECT(INDEX:INTEGER):TRECT
1712: Function ItemWidth( Index : Integer ) : Integer
1713: Function JavahashCode(val: string): Integer;
1714: Function JosephusG(n,k: integer; var graphout: string): integer;
1715: Function JulianDateToDateTime( const AValue : Double ) : TDateTime
1716: Function JvMessageBox( const Text, Caption : string; Flags : DWORD ) : Integer;
1717: Function JvMessageBox1( const Text : string; Flags : DWORD ) : Integer;
1718: Function KeepAlive : Boolean
1719: Function KeysToShiftState(Keys: Word): TShiftState;
1720: Function KeyDataToShiftState(KeyData: Longint): TShiftState;
1721: Function KeyboardStateToShiftState2(const KeyboardState: TKeyboardState): TShiftState;
1722: Function KeyboardStateToShiftState: TShiftState; overload;
1723: Function Languages : TLanguages
1724: Function Last : TClass
1725: Function Last : TComponent
1726: Function Last : TObject
1727: Function LastDelimiter( Delimiters, S : string ) : Integer
1728: Function LastDelimiter(const Delimiters: string; const S: string): Integer
1729: Function LastPos( const ASubstr : string; const AStr : string ) : Integer
1730: Function Latitude2WGS84(lat: double): double;
1731: Function LCM(m,n:longint):longint;
1732: Function LCMJ( const X, Y : Cardinal ) : Cardinal
1733: Function Ldexp( const X : Extended; const P : Integer ) : Extended
1734: Function LeftStr( const AText : AnsiString; const ACount : Integer ) : AnsiString;
1735: Function LeftStr( const AText : WideString; const ACount : Integer ) : WideString;
1736: function Length: Integer;
1737: Procedure LetFileList(FileList: TStringlist; apath: string);
1738: function lengthmp3(mp3path: string):integer;
1739: Function LineInRect( const P1, P2 : TPoint; const Rect : TRect ) : Boolean;
1740: Function LineInRect1( const P1, P2 : TFloatPoint; const Rect : TFloatRect ) : Boolean;
1741: Function LineSegmentIntersection(const L1P1 : TFloatPoint; L1P2 : TFloatPoint; const L2P1 : TFloatPoint; L2P2 : TFloatPoint; var P : TFloatPoint) : Boolean
1742: function LineStart(Buffer, BufPos: PChar): PChar
1743: function LineStart(Buffer, BufPos: PChar): PChar
1744: function ListSeparator: char;
1745: function Ln(x: Extended): Extended;
1746: Function LnXP1( const X : Extended ) : Extended
1747: function Lo(vdat: word): byte;
1748: Function LoadCursor(hInstance: HINST; lpCursorName: PChar): HCURSOR
1749: Function LoadedModulesList( const List : TStringList; ProcessID : DWORD; HandlesOnly : Boolean ) : Boolean

```



```

1750: Function LoadFileAsString( const FileName : string ) : string
1751: Function LoadFromFile( const FileName : string ) : TBitmapLoader
1752: Function LoadLibraryEx( LibName: PChar; hFile: Longint; Flags: Longint ): Longint; stdcall;
1753: Function LoadPackage( const Name: string ) : HMODULE
1754: Function LoadResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle ) : HGLOBAL
1755: Function LoadStr( Ident : Integer ) : string
1756: Function LoadString( Instance: Longint; IDent: Integer; Buffer: PChar; Size: Integer ): Integer; stdcall;
1757: Function LoadWideStr( Ident : Integer ) : WideString
1758: Function LOCATE( const KEYFIELDS: String; const KEYVALUES: VARIANT; OPTIONS: TLOCATEOPTIONS ) : BOOLEAN
1759: Function LockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint ) : HRESULT
1760: Function LockServer( fLock : LongBool ) : HRESULT
1761: Function LockVolume( const Volume : string; var Handle : THandle ) : Boolean
1762: Function Log( const X : Extended ) : Extended
1763: Function Log10( const X : Extended ) : Extended
1764: Function Log2( const X : Extended ) : Extended
1765: function LogBase10( X: Float ): Float;
1766: Function LogBase2( X: Float ): Float;
1767: Function LogBaseN( Base, X: Float ): Float;
1768: Function LogN( const Base, X : Extended ) : Extended
1769: Function LogOffOS : Boolean
1770: Function LoginDialog( const ADatabaseName : string; var AUserName, APassword : string ) : Boolean
1771: Function LoginDialogEx( const ADatabaseName: string; var AUserName, APassword: string; NameReadOnly: Bool ): Bool;
1772: Function LongDateFormat: string;
1773: function LongTimeFormat: string;
1774: Function LongWordToFourChar( ACardinal : LongWord ) : string
1775: Function LOOKUP( const KEYFIELDS: String; const KEYVALUES: VARIANT; const RESULTFIELDS: String ): VARIANT
1776: Function LookupName( const name : string ) : TInAddr
1777: Function LookupService( const service : string ) : Integer
1778: function Low: Int64;
1779: Function LowerCase( S : string ) : string
1780: Function Lowercase( s : AnyString ) : AnyString;
1781: Function LRot( const Value : Byte; const Count : TBitRange ) : Byte;
1782: Function LRotl( const Value : Word; const Count : TBitRange ) : Word;
1783: Function LRot2( const Value : Integer; const Count : TBitRange ) : Integer;
1784: function MainInstance: longword
1785: function MainThreadID: longword
1786: Function Map( x, in_min, in_max, out_min, out_max: integer ): integer; //arduino
1787: Function mapMax( ax, in_min, in_max, out_min, out_max: integer ): integer;
1788: Function MakeCanonicalIPv4Address( const AAddr : string ) : string
1789: Function MakeCanonicalIPv6Address( const AAddr : string ) : string
1790: Function MakeDIB( out Bitmap : PBitmapInfo ) : Integer
1791: Function MakeDWordIntoIPv4Address( const ADWord : Cardinal ) : string
1792: function MakeLong( A, B: Word ): Longint
1793: Function MakeTempFilename( const APath : String ) : string
1794: Function MakeValidFileName( const Str : string ) : string
1795: Function MakeValueMap( Enumeration : string; ToCds : Boolean ) : string
1796: function MakeWord( A, B: Byte ): Word
1797: Function MakeYear4Digit( Year, Pivot : Integer ) : Integer
1798: Function MapDateTime( const DateFormatType: string; DateFormat: string; Value: string; ToCds: Boolean ): string
1799: Function MapValues( Mapping : string; Value : string ) : string
1800: Function MaskDoFormatText( const EditMask : string; const Value : string; Blank : Char ) : string
1801: Function MaskGetCharType( const EditMask : string; MaskOffset : Integer ) : TMaskCharType
1802: Function MaskGetCurrentDirectives( const EditMask : string; MaskOffset : Integer ) : TMaskDirectives
1803: Function MaskGetFldSeparator( const EditMask : string ) : Integer
1804: Function MaskGetMaskBlank( const EditMask : string ) : Char
1805: Function MaskGetMaskSave( const EditMask : string ) : Boolean
1806: Function MaskIntlLiteralToChar( IChar : Char ) : Char
1807: Function MaskOffsetToOffset( const EditMask : string; MaskOffset : Integer ) : Integer
1808: Function MaskOffsetToWideOffset( const EditMask : string; MaskOffset : Integer ) : Integer
1809: Function MaskString( Mask, Value : string ) : string
1810: Function Match( const sString : string ) : TniRegularExpressionMatchResul
1811: Function Match1( const sString : string; iStart : integer ) : TniRegularExpressionMatchResult
1812: Function Matches( const Filename : string ) : Boolean
1813: Function MatchesMask( const Filename, Mask : string ) : Boolean
1814: Function MatchStr( const AText : string; const AValues : array of string ) : Boolean
1815: Function MatchText( const AText : string; const AValues : array of string ) : Boolean
1816: Function Max( AValueOne, AValueTwo : Integer ) : Integer
1817: function Max( const x, y: Integer ): Integer;
1818: Function Max1( const B1, B2 : Shortint ) : Shortint;
1819: Function Max2( const B1, B2 : Smallint ) : Smallint;
1820: Function Max3( const B1, B2 : Word ) : Word;
1821: function Max3( const x, y, z: Integer ): Integer;
1822: Function Max4( const B1, B2 : Integer ) : Integer;
1823: Function Max5( const B1, B2 : Cardinal ) : Cardinal;
1824: Function Max6( const B1, B2 : Int64 ) : Int64;
1825: Function Max64( const AValueOne, AValueTwo : Int64 ) : Int64
1826: Function MaxFloat( const X, Y : Float ) : Float
1827: Function MaxFloatArray( const B : TDynFloatArray ) : Float
1828: Function MaxFloatArrayIndex( const B : TDynFloatArray ) : Integer
1829: function MaxIntValue( const Data: array of Integer ): Integer
1830: Function MaxJ( const B1, B2 : Byte ) : Byte;
1831: function MaxPath: string;
1832: function MaxValue( const Data: array of Double ): Double
1833: Function MaxCalc( const Formula : string ) : Extended //math expression parser
1834: Procedure MaxCalcF( const Formula : string ); //out to console memo2
1835: function MD5( const fileName: string ): string;
1836: Function Mean( const Data : array of Double ) : Extended
1837: Function Median( const X : TDynFloatArray ) : Float
1838: Function Memory : Pointer

```

```

1839: Function MemoryPos( const ASubStr : string; MemBuff : PChar; MemorySize : Integer ) : Integer
1840: Function MessageBox(hndl: cardinal; text, caption: string; utype: cardinal): Integer;
1841: function MESSAGEBOX(TEXT,CAPTION:PCHAR;FLAGS:WORD):INTEGER
1842: Function MessageDlg(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
1843: Function MessageDlg1(const
Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;DefaultButton:TMsgDlgBtn): Integer;
1844: Function MessageDlgPos(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,
Y:Integer):Integer;
1845: Function MessageDlgPos1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn ) : Integer;
1846: Function MessageDlgPosHelp( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
Longint; X, Y : Integer; const HelpFileName : string ) : Integer;
1847: Function MessageDlgPosHelp1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx
: Longint; X, Y : Integer; const HelpFileName : string; DefaultButton : TMsgDlgBtn ) : Integer;
1848: Function MibToId( Mib : string ) : string
1849: Function MidStr( const AText : AnsiString; const AStart, ACount : Integer ) : AnsiString;
1850: Function MidStr( const AText : WideString; const AStart, ACount : Integer ) : WideString;
1851: Function microsecondsToCentimeters(mseconds: longint): longint; //340m/s speed of sound
1852: Function Micros(const Timer:THPTimer;const TimerRunning:Boolean):Int64//TypeS('THPTimer', 'Int64
1853: Function MIDIOut( DeviceID : Cardinal ) : IJclMIDIOut
1854: Procedure GetMidiOutputs( const List : TStrings)
1855: Function MIDISingleNoteTuningData( Key : TMIDINote; Frequency : Single ) : TSingleNoteTuningData
1856: Function MIDINoteToStr( Note : TMIDINote ) : string
1857: Function WinMidiOut( DeviceID : Cardinal ) : IJclWinMidiOut
1858: Procedure GetMidiOutputs( const List : TStrings)
1859: Procedure MidiOutCheck( Code : MMResult)
1860: Procedure MidiInCheck( Code : MMResult)
1861: Function MilliSecondOf( const AValue : TDateTime ) : Word
1862: Function MilliSecondOfDay( const AValue : TDateTime ) : LongWord
1863: Function MilliSecondOfTheHour( const AValue : TDateTime ) : LongWord
1864: Function MilliSecondOfTheMinute( const AValue : TDateTime ) : LongWord
1865: Function MilliSecondOfTheMonth( const AValue : TDateTime ) : LongWord
1866: Function MilliSecondOfTheSecond( const AValue : TDateTime ) : Word
1867: Function MilliSecondOfTheWeek( const AValue : TDateTime ) : LongWord
1868: Function MilliSecondOfTheYear( const AValue : TDateTime ) : Int64
1869: Function MilliSecondsBetween( const ANow, AThen : TDateTime ) : Int64
1870: Function MilliSecondSpan( const ANow, AThen : TDateTime ) : Double
1871: Function Micros( const Timer : THPTimer; const TimerRunning : Boolean ) : Int64
1872: Function millis: int64;
1873: Function Min( AValueOne, AValueTwo : Integer ) : Integer
1874: Function Min1( const B1, B2 : Shortint ) : Shortint;
1875: Function Min2( const B1, B2 : Smallint ) : Smallint;
1876: Function Min3( const B1, B2 : Word ) : Word;
1877: Function Min4( const B1, B2 : Integer ) : Integer;
1878: Function Min5( const B1, B2 : Cardinal ) : Cardinal;
1879: Function Min6( const B1, B2 : Int64 ) : Int64;
1880: Function Min64( const AValueOne, AValueTwo : Int64 ) : Int64
1881: Function MinClientRect : TRect;
1882: Function MinClientRect1( IncludeScroller : Boolean ) : TRect;
1883: Function MinClientRect2( TabCount : Integer; IncludeScroller : Boolean ) : TRect;
1884: Function MinFloat( const X, Y : Float ) : Float
1885: Function MinFloatArray( const B : TDynFloatArray ) : Float
1886: Function MinFloatArrayIndex( const B : TDynFloatArray ) : Integer
1887: Function MinimizeName( const Filename : string; Canvas : TCanvas; MaxLen : Integer ) : string
1888: Function MinimizeName( const Filename : TFileName; Canvas : TCanvas; MaxLen : Integer ) : TFileName
1889: Function MinimizeName(const Filename: string; Canvas: TCanvas;MaxLen: Integer): TFileName
1890: Function MinIntValue( const Data : array of Integer ) : Integer
1891: function MinIntValue(const Data: array of Integer):Integer)
1892: Function MinJ( const B1, B2 : Byte ) : Byte;
1893: Function MinuteOf( const AValue : TDateTime ) : Word
1894: Function MinuteOfDay( const AValue : TDateTime ) : Word
1895: Function MinuteOfTheHour( const AValue : TDateTime ) : Word
1896: Function MinuteOfTheMonth( const AValue : TDateTime ) : Word
1897: Function MinuteOfTheWeek( const AValue : TDateTime ) : Word
1898: Function MinuteOfTheYear( const AValue : TDateTime ) : LongWord
1899: Function MinutesBetween( const ANow, AThen : TDateTime ) : Int64
1900: Function MinuteSpan( const ANow, AThen : TDateTime ) : Double
1901: Function MinValue( const Data : array of Double ) : Double
1902: function MinValue(const Data: array of Double): Double)
1903: Function MixerLeftRightToArray( Left, Right : Cardinal ) : TDynCardinalArray
1904: Function MMCheck( const MciError : MCIERROR; const Msg : string ) : MCIERROR
1905: Function ModFloat( const X, Y : Float ) : Float
1906: Function ModifiedJulianDateToDateTime( const AValue : Double ) : TDateTime
1907: Function Modify( const Key : string; Value : Integer ) : Boolean
1908: Function ModuleCacheID : Cardinal
1909: Function ModuleFromAddr( const Addr : Pointer ) : HMODULE
1910: Function MonitorFromPoint( const Point : TPoint; MonitorDefault : TMonitorDefaultTo ) : TMonitor
1911: Function MonitorFromRect( const Rect : TRect; MonitorDefault : TMonitorDefaultTo ) : TMonitor
1912: Function MonitorFromWindow( const Handle : THandle; MonitorDefault : TMonitorDefaultTo ) : TMonitor
1913: Function MonthOf( const AValue : TDateTime ) : Word
1914: Function MonthOfTheYear( const AValue : TDateTime ) : Word
1915: Function MonthsBetween( const ANow, AThen : TDateTime ) : Integer
1916: Function MonthSpan( const ANow, AThen : TDateTime ) : Double
1917: Function MonthStr( DateTime : TDateTime ) : string
1918: Function MouseCoord( X, Y : Integer ) : TGridCoord
1919: Function MOVEBY( DISTANCE : INTEGER ) : INTEGER
1920: Function MoveFile( Source, Dest : string; Flags : FILEOP_FLAGS ) : Boolean
1921: Function MoveNext : Boolean
1922: Function MSecsToTimeStamp( MSecs : Comp ) : TTimeStamp

```

```

1923: function MSecsToTimeStamp(MSecs: Comp): TTimeStamp)
1924: Function Name : string
1925: Function NetPresentValue(const Rate:Extended;const CashFlows:array of
Double;PaymentTime:TPaymentTime):Extended
1926: function NetworkVolume(DriveChar: Char): string
1927: Function NEWBOTOMLINE : INTEGER
1928: Function NewCompareNode( Field : TField; Operator : TCANOperator; const Value : Variant ) : PExprNode
1929: Function NEWITEM( const ACAPTION : String; ASHORTCUT : TSHORTCUT; ACHECKED, AENABLED : BOOLEAN; AONCLICK :
TNOTIFYEVENT; HCTX : WORD; const ANAME : String ) : TMENUITEM
1930: Function NEWLINE : TMENUITEM
1931: Function NEWMENU( OWNER : TCOMPONENT; const ANAME : STRING; ITEMS : array of TMenuItem ) : TMAINMENU
1932: Function NewNode(Kind: TExprNodeKind; Operator:TCANOperator; const Data:Variant; Left,
Right:PExprNode):PExprNode
1933: Function NEWPOPUPMENU(OWNER:TCOMPONENT;const ANAME:String; ALIGNMENT:TPOPOPALIGNMENT;AUTOPOPUP:BOOLEAN;
const ITEMS : array of TCMENUITEM ) : TPOPUPMENU
1934: Function NewState( eType : TniRegularExpressionStateType ) : TniRegularExpressionState
1935: Function NEWSUBMENU(const ACAPT:String;HCTX:WORD;const ANAME:String;ITEMS:array of
TMenuItem;AENABLED:BOOL): TMENUITEM
1936: Function NEWTOPLINE : INTEGER
1937: Function Next : TIdAuthWhatsNext
1938: Function NextCharIndex( S : String; Index : Integer ) : Integer
1939: Function NextRecordSet : TCustomSQLDataSet
1940: Function NextRecordset( var RecordsAffected : Integer ) : _Recordset
1941: Function NextSQLToken1( var p : WideChar; out Token : WideString; CurSection : TSQLToken ) : TSQLToken;
1942: Function NextToken : Char
1943: Function nextToken : WideString
1944: function NextToken:Char
1945: Function Norm( const Data : array of Double ) : Extended
1946: Function NormalizeAngle( const Angle : Extended ) : Extended
1947: Function NormalizeBcd( const InBcd : TBcd; var OutBcd : TBcd; const Prec, Scale : Word ) : Boolean
1948: Function NormalizeRect( const Rect : TRect ) : TRect
1949: function NormalizeRect(const Rect: TRect): TRect;
1950: Function Now : TDateTime
1951: function Now2: tDateTime
1952: Function NumProcessThreads : integer
1953: Function NumThreadCount : integer
1954: Function NthDayOfWeek( const AValue : TDateTime ) : Word
1955: Function NtProductType : TntProductType
1956: Function NtProductTypeString : string
1957: function Null: Variant;
1958: Function NullPoint : TPoint
1959: Function NullRect : TRect
1960: Function Null2Blank(aString:String):String;
1961: Function NumberOfPeriods( const Rate : Extended; Payment : Extended; const PresentValue, FutureValue :
Extended; PaymentTime : TPaymentTime ) : Extended
1962: Function NumIP : integer
1963: function Odd(x: Longint): boolean;
1964: Function OffsetFromUTC : TDateTime
1965: Function OffsetPoint( const P, Offset : TPoint ) : TPoint
1966: Function OffsetRect( var Rect : TRect; DX : Integer; DY : Integer ) : Boolean
1967: function OffsetRect(var Rect: TRect; DX:Integer; DY:Integer): Boolean)
1968: Function OffsetToMaskOffset( const EditMask : string; Offset : Integer ) : Integer
1969: Function OkToChangeFieldAlignment( AField : TField; Alignment : TAlignment ) : Boolean
1970: Function OldBCDToCurr( const BCD : TBcd; var Curr : Currency ) : Boolean
1971: Function OldCurrToBCD(const Curr:Currency; var BCD:TBcd; Precision:Integer;Decimals:Integer): Boolean
1972: function OpenBit:Integer
1973: Function OpenDatabase : TDatabase
1974: Function OpenDatabase( const DatabaseName : string ) : TDatabase
1975: Function OpenGLColorToWinColor( const Red, Green, Blue : Float ) : TColor
1976: Function OpenObject( Value : PChar ) : Boolean;
1977: Function OpenObject1( Value : string ) : Boolean;
1978: Function OpenSession( const SessionName : string ) : TSession
1979: Function OpenVolume( const Drive : Char ) : THandle
1980: function OrdFourByteToCardinal(ABYTE1, AByte2, AByte3, AByte4 : Byte): Cardinal
1981: Function OrdFourByteToLongWord( AByte1, AByte2, AByte3, AByte4 : Byte ) : LongWord
1982: Function OrdToBinary( const Value : Byte ) : string;
1983: Function OrdToBinary1( const Value : Shortint ) : string;
1984: Function OrdToBinary2( const Value : Smallint ) : string;
1985: Function OrdToBinary3( const Value : Word ) : string;
1986: Function OrdToBinary4( const Value : Integer ) : string;
1987: Function OrdToBinary5( const Value : Cardinal ) : string;
1988: Function OrdToBinary6( const Value : Int64 ) : string;
1989: Function OSCheck( RetVal : Boolean ) : Boolean
1990: Function OSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD ) : string
1991: Function OSIdentToString( const OSIdent : DWORD ) : string
1992: Function Output( Text )
1993: Function Overlay( ImageIndex : Integer; Overlay : TOverlay ) : Boolean
1994: Function Owner : TCustomListView
1995: function Owner : TPersistent
1996: Function PadInputLiterals( const EditMask : String; const Value : string; Blank : Char ) : string
1997: Function PadL( pStr : String; pLth : integer ) : String
1998: Function Padl(s : AnyString;I : longInt) : AnyString;
1999: Function PadLCh( pStr : String; pLth : integer; pChr : char ) : String
2000: Function PadR( pStr : String; pLth : integer ) : String
2001: Function Padr(s : AnyString;I : longInt) : AnyString;
2002: Function PadRCh( pStr : String; pLth : integer; pChr : char ) : String
2003: Function PadString( const AString : String; const ALen : Integer; const AChar : Char ) : String
2004: Function Padz(s : AnyString;I : longInt) : AnyString;
2005: Function PaethPredictor( a, b, c : Byte ) : Byte

```

```

2006: Function PARAMBYNAME( const VALUE : String ) : TPARAM
2007: Function ParamByName( const Value : WideString ) : TParameter
2008: Function ParamCount: Integer
2009: Function ParamsEncode( const ASrc : string ) : string
2010: function ParamStr(Index: Integer): string
2011: Function ParseDate( const DateStr : string ) : TDateTime
2012: Function PARSESQL( SQL : String; DOCREATE : BOOLEAN ) : String
2013: Function ParseSQL( SQL : WideString; DoCreate : Boolean ) : WideString
2014: Function PathAddExtension( const Path, Extension : string ) : string
2015: Function PathAddSeparator( const Path : string ) : string
2016: Function PathAppend( const Path, Append : string ) : string
2017: Function PathBuildRoot( const Drive : Byte ) : string
2018: Function PathCanonicalize( const Path : string ) : string
2019: Function PathCommonPrefix( const Path1, Path2 : string ) : Integer
2020: Function PathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string;
2021: Function PathCompactPath1(const Canv:TCanvas;const Path:string;const Width:Int;CmpFmt:TCompactPath):string;
2022: Function PathEncode( const ASrc : string ) : string
2023: Function PathExtractFileDirFixed( const S : AnsiString ) : AnsiString
2024: Function PathExtractFileNameNoExt( const Path : string ) : string
2025: Function PathExtractPathDepth( const Path : string; Depth : Integer ) : string
2026: Function PathGetDepth( const Path : string ) : Integer
2027: Function PathGetLongName( const Path : string ) : string
2028: Function PathGetLongName2( Path : string ) : string
2029: Function PathGetShortName( const Path : string ) : string
2030: Function PathIsAbsolute( const Path : string ) : Boolean
2031: Function PathIsChild( const Path, Base : AnsiString ) : Boolean
2032: Function PathIsDiskDevice( const Path : string ) : Boolean
2033: Function PathIsUNC( const Path : string ) : Boolean
2034: Function PathRemoveExtension( const Path : string ) : string
2035: Function PathRemoveSeparator( const Path : string ) : string
2036: Function Payment(Rate:Extended;NPeriods:Int;const PresentVal,
FutureVal:Extended;PaymentTime:TPaymentTime):Extended
2037: Function Peek : Pointer
2038: Function Peek : TObject
2039: function PERFORM(MSG:CARDINAL;WPARAM,LPARAM:LONGINT):LONGINT
2040: Function PeriodPayment(const Rate:Extended;Period,NPeriods:Integer; const PresentValue, FutureValue :
Extended; PaymentTime : TPaymentTime ) : Extended
2041: function Permutation(npr, k: integer): extended;
2042: function PermutationInt(npr, k: integer): Int64;
2043: Function PermutationJ( N, R : Cardinal ) : Float
2044: Function Pi : Extended;
2045: Function PiE : Extended;
2046: Function PixelsToDialogsX( const Pixels : Word ) : Word
2047: Function PixelsToDialogsY( const Pixels : Word ) : Word
2048: Function PlaySound(s: pchar; flag,syncflag: integer): boolean;
2049: Function Point( X, Y : Integer ) : TPoint
2050: function Point(X, Y: Integer): TPoint)
2051: Function PointAssign( const X, Y : Integer ) : TPoint
2052: Function PointDist( const P1, P2 : TPoint ) : Double;
2053: function PointDist(const P1,P2: TFloatPoint): Double;
2054: Function PointDist1( const P1, P2 : TFloatPoint ) : Double;
2055: function PointDist2(const P1,P2: TPoint): Double;
2056: Function PointEqual( const P1, P2 : TPoint ) : Boolean
2057: Function PointIsNull( const P : TPoint ) : Boolean
2058: Function PointToLineSegmentDist( const Point, LineP1, LineP2 : TFloatPoint ) : Double
2059: Function Poly( const X : Extended; const Coefficients : array of Double ) : Extended
2060: Function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
2061: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
2062: Function Pop : Pointer
2063: Function Pop : TObject
2064: Function PopnStdDev( const Data : array of Double ) : Extended
2065: Function PopnVariance( const Data : array of Double ) : Extended
2066: Function PopulationVariance( const X : TDynFloatArray ) : Float
2067: function Pos(SubStr, S: AnyString): Longint;
2068: Function PosEqual( const Rect : TRect ) : Boolean
2069: Function PosEx( const SubStr, S : string; Offset : Integer ) : Integer
2070: Function PosInSmallIntArray( const ASearchInt : SmallInt; AArray : array of SmallInt ) : Integer
2071: Function PosInStrArray(const SearchStr:string;Contents:array of string;const CaseSensitive:Boolean):Integer
2072: Function Post1( AURL : string; const ASource : TStrings ) : string;
2073: Function Post2( AURL : string; const ASource : TStream ) : string;
2074: Function Post3( AURL : string; const ASource : TIdMultiPartFormDataStream ) : string;
2075: Function PostData( const UserData : WideString; const CheckSum : DWORD ) : Boolean
2076: Function PostData( const UserData : WideString; const CheckSum : integer ) : Boolean
2077: function PostMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2078: Function Power( const Base, Exponent : Extended ) : Extended
2079: Function PowerBig(aval, n:integer): string;
2080: Function PowerIntJ( const X : Float; N : Integer ) : Float;
2081: Function PowerJ( const Base, Exponent : Float ) : Float;
2082: Function PowerOffOS : Boolean
2083: Function PreformatDateString( Ps : string ) : string
2084: Function PresentValue(const Rate:Extend;NPeriods:Int;const Payment,
FutureVal:Extend;PaymentTime:TPaymentTime):Extended
2085: Function PrimeFactors( N : Cardinal ) : TDynCardinalArray
2086: Function Printer : TPrinter
2087: Function ProcessPath2( const ABasePath:String; const APath: String; const APathDelim:string) : string
2088: Function ProcessResponse : TIdHTTPWhatsNext
2089: Function ProduceContent : string
2090: Function ProduceContentFromStream( Stream : TStream ) : string
2091: Function ProduceContentFromString( const S : string ) : string

```



```

2092: Function ProgIDToClassID(const ProgID: string): TGUID;
2093: Function PromptDataLinkFile( ParentHandle : THandle; InitialFile : WideString) : WideString
2094: Function PromptDataSource( ParentHandle : THandle; InitialString : WideString) : WideString
2095: Function PromptForFileName( var AFileName : string; const AFilter : string; const ADefaultExt : string;
const ATitle : string; const AInitialDir : string; SaveDialog : Boolean) : Boolean
2096: function PromptForFileName(var AFileName: string; const AFilter: string; const ADefaultExt: string; const
ATitle: string; const AInitialDir: string; SaveDialog: Boolean): Boolean)
2097: Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FileName,Output:String):Boolean
2098: Function PtInRect( const Rect : TRect; const P : TPoint) : Boolean
2099: function PtInRect(const Rect: TRect; const P: TPoint): Boolean)
2100: Function Push( AItem : Pointer) : Pointer
2101: Function Push( AObject : TObject) : TObject
2102: Function Put1( AURL : string; const ASource : TStream) : string;
2103: Function Pythagoras( const X, Y : Extended) : Extended
2104: Function queryDLLInterface( var queryList : TStringList) : TStringList
2105: Function queryDLLInterfaceTwo( var queryList : TStringList) : TStringList
2106: Function QueryInterface(const IID: TGUID; out Obj): HRESULT, CdStdCall
2107: Function queryPerformanceCounter2(mse: int64): int64;
2108: //Function QueryPerformanceCounter(var lpPerformanceCount: Int64): LongBool; stdcall;
2109: //Function QueryPerformanceFrequency(mse: int64): boolean;
2110: Function QueryPerformanceCounter(var lcount: Int64): Boolean; stdcall;
2111: Function QueryPerformanceFrequency(var lfreq: int64): boolean; stdcall;
2112: Procedure QueryPerformanceCounter1(var aC: Int64);
2113: Function QueryPerformanceFrequency1(var freq: int64): boolean;
2114: Function Quote( const ACommand : String) : SmallInt
2115: Function QuotedStr( S : string) : string
2116: Function RadToCycle( const Radians : Extended) : Extended
2117: Function RadToDeg( const Radians : Extended) : Extended
2118: Function RadToDeg( const Value : Extended) : Extended;
2119: Function RadToDeg1( const Value : Double) : Double;
2120: Function RadToDeg2( const Value : Single) : Single;
2121: Function RadToGrad( const Radians : Extended) : Extended
2122: Function RadToGrad( const Value : Extended) : Extended;
2123: Function RadToGrad1( const Value : Double) : Double;
2124: Function RadToGrad2( const Value : Single) : Single;
2125: Function RandG( Mean, StdDev : Extended) : Extended
2126: function Random(const ARange: Integer): Integer;
2127: function random2(a: integer): double
2128: function RandomE: Extended;
2129: function RandomF: Extended;
2130: Function RandomFrom( const AValues : array of string) : string;
2131: Function RandomRange( const AFrom, ATo : Integer) : Integer
2132: function randSeed: longint
2133: Function RawToDataColumn( ACol : Integer) : Integer
2134: Function Read : Char
2135: Function Read( pv : Pointer; cb : Longint; pcbRead : PLongint) : HRESULT
2136: function Read(Buffer:String;Count:LongInt):LongInt
2137: Function ReadBinaryStream( const Section, Name : string; Value : TStream) : Integer
2138: Function ReadBool( const Section, Ident : string; Default : Boolean) : Boolean
2139: Function ReadCardinal( const AConvert : boolean) : Cardinal
2140: Function ReadChar : Char
2141: Function ReadClient( var Buffer, Count : Integer) : Integer
2142: Function ReadDate( const Section, Name : string; Default : TDateTime) : TDateTime
2143: Function ReadDateTime( const Section, Name : string; Default : TDateTime) : TDateTime
2144: Function ReadFloat( const Section, Name : string; Default : Double) : Double
2145: Function ReadFromStack(const ARaiseExceptfDisconnected:Bool;ATimeout:Int;const ARaiseExceptonTimeout:
Boolean):Integer
2146: Function ReadInteger( const AConvert : boolean) : Integer
2147: Function ReadInteger( const Section, Ident : string; Default : Longint) : Longint
2148: Function ReadLn : string
2149: Function ReadLn( ATerminator : string; const ATimeout : Integer; AMaxLineLength : Integer) : string
2150: function ReadLn(question: string): string;
2151: Function ReadLnWait( AFailCount : Integer) : string
2152: Function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
2153: Function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
2154: Function ReadSmallInt( const AConvert : boolean) : SmallInt
2155: Function ReadString( const ABytes : Integer) : string
2156: Function ReadString( const Section, Ident, Default : string) : string
2157: Function ReadString( Count : Integer) : string
2158: Function ReadTime( const Section, Name : string; Default : TDateTime) : TDateTime
2159: Function ReadTimeStampCounter : Int64
2160: Function RebootOS : Boolean
2161: Function Receive( ATimeout : Integer) : TReplyStatus
2162: Function ReceiveBuf( var Buf, Count : Integer) : Integer
2163: Function ReceiveLength : Integer
2164: Function ReceiveText : string
2165: Function ReceiveSerialData(var Data: TByteArray; DataSize: cardinal): cardinal
2166: Function ReceiveSerialText: string
2167: Function RecodeDate( const AValue : TDateTime; const AYear, AMonth, ADay : Word) : TDateTime
2168: Function RecodeDateTime(const AValue:TDateTime;const AYear,AMonth,ADay,AHr,AMin,ASec,
AMilliSec:Word):TDateTime
2169: Function RecodeDay( const AValue : TDateTime; const ADay : Word) : TDateTime
2170: Function RecodeHour( const AValue : TDateTime; const AHour : Word) : TDateTime
2171: Function RecodeMilliSecond( const AValue : TDateTime; const AMilliSecond : Word) : TDateTime
2172: Function RecodeMinute( const AValue : TDateTime; const AMinute : Word) : TDateTime
2173: Function RecodeMonth( const AValue : TDateTime; const AMonth : Word) : TDateTime
2174: Function RecodeSecond( const AValue : TDateTime; const ASecond : Word) : TDateTime
2175: Function RecodeTime( const AValue : TDateTime;const AHour,AMinute,ASecond,AMilliSecond:Word):TDateTime
2176: Function RecodeYear( const AValue : TDateTime; const AYear : Word) : TDateTime

```

```

2177: Function Reconcile( const Results : OleVariant) : Boolean
2178: Function Rect( Left, Top, Right, Bottom : Integer) : TRect
2179: function Rect(ALeft: Integer; ATop: Integer; ARight: Integer; ABottom: Integer): TRect)
2180: Function Rect2( const ATopLeft, ABottomRight : TPoint) : TRect;
2181: Function RectAssign( const Left, Top, Right, Bottom : Integer) : TRect
2182: Function RectAssignPoints( const TopLeft, BottomRight : TPoint) : TRect
2183: Function RectBounds( const Left, Top, Width, Height : Integer) : TRect
2184: Function RectCenter( const R : TRect) : TPoint
2185: Function RectEqual( const R1, R2 : TRect) : Boolean
2186: Function RectHeight( const R : TRect) : Integer
2187: Function RectIncludesPoint( const R : TRect; const Pt : TPoint) : Boolean
2188: Function RectIncludesRect( const R1, R2 : TRect) : Boolean
2189: Function RectIntersection( const R1, R2 : TRect) : TRect
2190: Function RectIntersectRect( const R1, R2 : TRect) : TRect
2191: Function RectIsEmpty( const R : TRect) : Boolean
2192: Function RectIsNull( const R : TRect) : Boolean
2193: Function RectIsSquare( const R : TRect) : Boolean
2194: Function RectIsValid( const R : TRect) : Boolean
2195: Function RectsAreValid( R : array of TRect) : Boolean
2196: Function RectUnion( const R1, R2 : TRect) : TRect
2197: Function RectWidth( const R : TRect) : Integer
2198: Function RedComponent( const Color32 : TColor32) : Integer
2199: Function Refresh : Boolean
2200: Function RefStringListCopy(aRefArray:TStringlist):TStringList;
2201: Function RegisterConversionFamily( const ADescription : string) : TConvFamily
2202: Function RegisterConversionType( AConvTypeInfo : TConvTypeInfo; out AType : TConvType) : Boolean;
2203: Function RegisterConversionType(const AFam:TConvFam;const ADescr:string;const AFact:Double):TConvType
2204: Function RegistryRead(keyHandle: Longint; keyPath, myField: String): string;
2205: Function ReleaseDC(hdwnd: HWND; hdc: HDC): integer;
2206: Function ReleaseHandle : HBITMAP
2207: Function ReleaseHandle : HENHMETAFILE
2208: Function ReleaseHandle : HICON
2209: Function ReleasePalette : HPALETTE
2210: Function RemainderFloat( const X, Y : Float) : Float
2211: Function Remove( AClass : TClass) : Integer
2212: Function Remove( AComponent : TComponent) : Integer
2213: Function Remove( AItem : Integer) : Integer
2214: Function Remove( AItem : Pointer) : Pointer
2215: Function Remove( Aitem : TObject) : TObject
2216: Function Remove( AObject : TObject) : Integer
2217: Function RemoveBackslash( const PathName : string) : string
2218: Function RemoveDF( aString : String) : String //removes thousand separator
2219: Function RemoveDir( Dir : string) : Boolean
2220: function RemoveDir(const Dir: string): Boolean)
2221: Function RemoveDirectory(PathName: PChar): WordBool; stdcall;
2222: Function RemoveFileExt( const FileName : string) : string
2223: Function RemoveHeaderEntry( AHeader, AEntry : string) : string
2224: Function RenameFile( OldName, NewName : string) : Boolean
2225: function RenameFile(const OldName: string; const NewName: string): Boolean)
2226: Function ReplaceStr( const AText, AFromText, AToText : string) : string
2227: Function ReplaceText( const AText, AFromText, AToText : string) : string
2228: Function Replicate(c : char;I : longInt) : String;
2229: Function Request : TWebRequest
2230: Function ResemblesText( const AText, AOther : string) : Boolean
2231: Function Reset : Boolean
2232: function Reset2(mypath: string):string;
2233: Function ResInstLoad(Instance:THandle;ResType:TResType; const Name:string;MaskColor: TColor): Boolean
2234: Function ResourceLoad( ResType : TResType; const Name : string; MaskColor : TColor) : Boolean
2235: Function Response : TWebResponse
2236: Function ResumeSupported : Boolean
2237: Function RETHINKHOTKEYS : BOOLEAN
2238: Function RETHINKLINES : BOOLEAN
2239: Function Retrieve( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2240: Function RetrieveCurrentDir : string
2241: Function RetrieveDeltas( const cdsArray : array of TClientDataset) : Variant
2242: Function RetrieveHeader( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2243: Function RetrieveMailBoxSize : integer
2244: Function RetrieveMsgSize( const MsgNum : Integer) : Integer
2245: Function RetrieveProviders( const cdsArray : array of TClientDataset) : Variant
2246: Function RetrieveRaw( const MsgNum : Integer; const Dest : TStrings) : boolean
2247: Function ReturnMIMEType( var MediaType, EncType : String) : Boolean
2248: Function ReverseBits( Value : Byte) : Byte;
2249: Function ReverseBits1( Value : Shortint) : Shortint;
2250: Function ReverseBits2( Value : Smallint) : Smallint;
2251: Function ReverseBits3( Value : Word) : Word;
2252: Function ReverseBits4( Value : Cardinal) : Cardinal;
2253: Function ReverseBits4( Value : Integer) : Integer;
2254: Function ReverseBits5( Value : Int64) : Int64;
2255: Function ReverseBytes( Value : Word) : Word;
2256: Function ReverseBytes1( Value : Smallint) : Smallint;
2257: Function ReverseBytes2( Value : Integer) : Integer;
2258: Function ReverseBytes3( Value : Cardinal) : Cardinal;
2259: Function ReverseBytes4( Value : Int64) : Int64;
2260: Function ReverseString( const AText : string) : string
2261: Function ReverseDNSLookup(const IPAddr:String;DNSServer:String;Timeout,Retries:Int;var
  HostName:String):Bool;
2262: Function Revert : HRESULT
2263: Function RGB(R,G,B: Byte): TColor;
2264: Function RGB2BGR( const Color : TColor) : TColor

```

```

2265: Function RGB2TColor( R, G, B : Byte ) : TColor
2266: Function RGBToWebColorName( RGB : Integer ) : string
2267: Function RGBToWebColorStr( RGB : Integer ) : string
2268: Function RgbToHtml( Value : TColor ) : string
2269: Function HtmlToRgb(const Value: string): TColor;
2270: Function RightStr( const AStr : String; Len : Integer ) : String
2271: Function RightStr( const AText : AnsiString; const ACount : Integer ) : AnsiString;
2272: Function RightStr( const AText : WideString; const ACount : Integer ) : WideString;
2273: Function ROL( AVal : LongWord; AShift : Byte ) : LongWord
2274: Function ROR( AVal : LongWord; AShift : Byte ) : LongWord
2275: Function RotatePoint( Point : TFloatPoint; const Center : TFloatPoint; const Angle : Float ) : TFloatPoint
2276: function RotatePoint(Point: TFloatPoint; const Center: TFloatPoint; const Angle: Double): TFloatPoint;
2277: Function Round(e : Extended) : Longint;
2278: Function Round64(e: extended): Int64;
2279: Function RoundAt( const Value : string; Position : SmallInt) : string
2280: Function RoundFrequency( const Frequency : Integer ) : Integer
2281: Function RoundInt( Value : Integer; StepSize : Integer ) : Integer
2282: Function RoundPoint( const X, Y : Double ) : TPoint
2283: Function RoundRect( const ALeft, ATop, ARight, ABottom : Double ) : TRect
2284: Function RowCount : Integer
2285: Function RowRequest( const Row : OleVariant; RequestType : Integer; var OwnerData : OleVariant): OleVariant
2286: Function RowRequest( Row : OleVariant; Options : TFetchOptions ) : OleVariant
2287: Function RPos( const ASub, AIn : String; AStart : Integer ) : Integer
2288: Function RRot( const Value : Byte; const Count : TBitRange ) : Byte;
2289: Function RRot1( const Value : Word; const Count : TBitRange ) : Word;
2290: Function RRot2( const Value : Integer; const Count : TBitRange ) : Integer;
2291: Function RunDLL32(const ModuleNa,FuncName,CmdLine:string;WaitForCompletion:Bool;CmdShow:Integer):Boolean
2292: Function RunningProcessesList( const List : TStringList; FullPath : Boolean ) : Boolean
2293: Function S_AddBackSlash( const ADirName : string ) : string
2294: Function S_AllTrim( const cStr : string ) : string
2295: Function S_AtRepl( const cAT, cStr, cRepl : string ) : string
2296: Function S_Cut( const cStr : string; const iLen : integer ) : string
2297: Function S_DecryptCRC32( const crc : string; StartKey, MultKey, AddKey : integer ) : integer
2298: Function S_DirExists( const ADir : string ) : Boolean
2299: Function S_Empty( const cStr : string ) : boolean
2300: Function S_EncryptCRC32( const crc : LongWORD; StartKey, MultKey, AddKey : integer ) : string
2301: Function S_LargeFontsActive : Boolean
2302: Function S_LimitDigits( AValue : Extended; ANumDigits : Integer ) : Extended
2303: Function S_LTrim( const cStr : string ) : string
2304: Function S_ReadNextTextLineFromStream( stream : TStream ) : string
2305: Function S_RepeatChar( const iLen : integer; const AChar : Char ) : String
2306: Function S_Rep1First( const cAT, cStr, cRepl : string ) : string
2307: Function S_RoundDecimal( AValue : Extended; APlaces : Integer ) : Extended
2308: Function S_RTrim( const cStr : string ) : string
2309: Function S_RTrimCopy( const cStr : string; iPos, iLen : integer ) : string
2310: //Type TS_ShellExecuteCmd = (seCmdOpen,seCmdPrint,seCmdExplore);
2311: Function S_ShellExecute( aFilename : string; aParameters : string; aCommand : TS_ShellExecuteCmd ) : string
2312: Function S_Space( const iLen : integer ) : String
2313: Function S_StrBlanks( const cStr : string; const iLen : integer ) : string
2314: Function S_StrBlanksCuttooLong( const cStr : string; const iLen : integer ) : string
2315: Function S_StrCRC32( const Text : string ) : LongWORD
2316: Function S_StrDecrypt96( const InString : string; StartKey, MultKey, AddKey : Integer ) : string
2317: Function S_StrEncrypt96( const InString : string; StartKey, MultKey, AddKey : Integer ) : string
2318: Function S_StringToUTF_8( const AString : string ) : string
2319: Function S_StrLBlanks( const cStr : string; const iLen : integer ) : string
2320: function S_StrToReal(const cStr: string; var R: Double): Boolean
2321: Function S_TokenEnd( cBuffer : PChar; lEmptyToken : boolean ) : boolean
2322: Function S_TokenNext( cBuffer : PChar; lEmptyToken : boolean ) : string
2323: Function S_UTF_8ToString( const AString : string ) : string
2324: Function S_WBox( const AText : string ) : integer
2325: Function SameDate( const A, B : TDateTime ) : Boolean
2326: function SameDate(const A, B: TDateTime): Boolean;
2327: Function SameDateTime( const A, B : TDateTime ) : Boolean
2328: function SameDateTime(const A, B: TDateTime): Boolean;
2329: Function SameFileName( S1, S2 : string ) : Boolean
2330: Function SameText( S1, S2 : string ) : Boolean
2331: Function SameText( const S1: string; const S2: string): Boolean)
2332: Function SameTime( const A, B : TDateTime ) : Boolean
2333: function SameTime(const A, B: TDateTime): Boolean;
2334: function SameValue(const A, B: Extended; Epsilon: Extended): Boolean //overload;
2335: function SameValue1(const A, B: Double; Epsilon: Double): Boolean //overload;
2336: function SameValue2(const A, B: Single; Epsilon: Single): Boolean //overload;
2337: Function SampleVariance( const X : TDynFloatArray ) : Float
2338: Function Sar( const Value : Shortint; const Count : TBitRange ) : Shortint;
2339: Function Sar1( const Value : Smallint; const Count : TBitRange ) : Smallint;
2340: Function Sar2( const Value : Integer; const Count : TBitRange ) : Integer;
2341: Function SaveToFile( const AFileName : TFileName ) : Boolean
2342: Function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
2343: Function SaveAsExcel(aGrid: TStringGrid; aSheetName, aFileName: string; openexcel: boolean): Boolean;
2344: Function ScanF(const aformat: String; const args: array of const): string;
2345: Function SCREENTOCLIENT(POINT:TPOINT):TPOINT
2346: Function SearchBuf( Buf: PChar; BufLen: Integer; SelStart, SelLength: Integer; SearchString: String; Options: TStringSearchOptions): PChar
2347: Function SearchBuf2( Buf: String; SelStart, SelLength: Integer; SearchString: String; Options: TStringSearchOptions): Integer;
2348: function SearchRecattr: integer;
2349: function SearchRecExcludeAttr: integer;
2350: Function SearchRecFileSize64( const SearchRec : TSearchRec ) : Int64
2351: function SearchRename: string;

```

```

2352: function SearchRecsize: integer;
2353: function SearchRecTime: integer;
2354: function Sec( const X : Extended) : Extended
2355: function Secant( const X : Extended) : Extended
2356: function SecH( const X : Extended) : Extended
2357: function SecondOf( const AValue : TDateTime) : Word
2358: function SecondOfDay( const AValue : TDateTime) : LongWord
2359: function SecondOfTheHour( const AValue : TDateTime) : Word
2360: function SecondOfTheMinute( const AValue : TDateTime) : Word
2361: function SecondOfTheMonth( const AValue : TDateTime) : LongWord
2362: function SecondOfTheWeek( const AValue : TDateTime) : LongWord
2363: function SecondOfTheYear( const AValue : TDateTime) : LongWord
2364: function SecondsBetween( const ANow, AThen : TDateTime) : Int64
2365: function SecondSpan( const ANow, AThen : TDateTime) : Double
2366: function SectionExists( const Section : string) : Boolean
2367: function Seek( const KeyValues : Variant; SeekOption : TSeekOption) : Boolean
2368: function Seek( dlibMove : Longint; dwOrigin : Longint; out libNewPosition : Largeint) : HRESULT
2369: function Seek(Offset:Longint;Origin:Word):Longint
2370: function SelectDirectory( var Directory : string; Options : TSelectDirOpts; HelpCtx:Longint): Boolean;
2371: function SelectDirectory1( const Caption : string; const Root : WideString; var Directory : string;
Options : TSelectDirExtOpts; Parent : TWinControl) : Boolean;
2372: function SelectImage( var AFileName : string; const Extensions, Filter : string) : Boolean
2373: function SendAppMessage(Msg: Cardinal; WParam, LParam: Longint): Longint
2374: function SendBuf( var Buf, Count : Integer) : Integer
2375: function SendCmd( const AOut : string; const AResponse : SmallInt) : SmallInt;
2376: function SendCmd1( const AOut : string; const AResponse : array of SmallInt) : SmallInt;
2377: function SendKey( AppName : string; Key : Char) : Boolean
2378: function SendMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2379: function SendStream( AStream : TStream) : Boolean
2380: function SendStreamThenDrop( AStream : TStream) : Boolean
2381: function SendText( const S : string) : Integer
2382: function SendSerialData(Data: TByteArray; DataSize: cardinal): cardinal
2383: function SendSerialText(Data: String): cardinal
2384: function Sent : Boolean
2385: function ServicesFilePath: string
2386: function SetAlpha( const Color32 : TColor32; NewAlpha : Integer) : TColor32
2387: function SetBit( const Value : Byte; const Bit : TBitRange) : Byte;
2388: function SetBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2389: function SetBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2390: function SetBit3( const Value : Word; const Bit : TBitRange) : Word;
2391: function SetBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2392: function SetBit4( const Value : Integer; const Bit : TBitRange) : Integer;
2393: function SetBit5( const Value : Int64; const Bit : TBitRange) : Int64;
2394: function SetClipboard( NewClipboard : TClipboard) : TClipboard
2395: function SetColorBlue( const Color : TColor; const Blue : Byte) : TColor
2396: function SetColorFlag( const Color : TColor; const Flag : Byte) : TColor
2397: function SetColorGreen( const Color : TColor; const Green : Byte) : TColor
2398: function SetColorRed( const Color : TColor; const Red : Byte) : TColor
2399: function SetCurrentDir( Dir : string) : Boolean
2400: function SetCurrentDir(const Dir: string): Boolean)
2401: function SetCurrentDirectory(PathName: PChar): WordBool; stdcall;
2402: function SetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
2403: function SetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
2404: function SetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
2405: function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
2406: function SetEndOfFile(Handle: Integer): LongBool; stdcall;
2407: function SetEnvironmentVar( const Name, Value : string) : Boolean
2408: function SetErrorProc( ErrorProc : TSocketErrorProc) : TSocketErrorProc
2409: function SetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
2410: function SetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
2411: function SetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
2412: function SetFileTimeStamp( const FileName : string; TimeStamp : Integer) : Boolean
2413: function SETFOCUSEDCONTROL(CONTROL:TWINCONTROL):BOOLEAN
2414: function SetLocalTime( Value : TDateTime) : boolean
2415: function SetPrecisionTolerance( NewTolerance : Float) : Float
2416: function SetPrinter( NewPrinter : TPrinter) : TPrinter
2417: function SetRGBValue( const Red, Green, Blue : Byte) : TColor
2418: function SetSequence( S, Localizar, Substituir : shortstring) : shortstring
2419: function SetSize( libNewSize : Longint) : HRESULT
2420: function SetUserObjectFullAccess( hUserObject : THandle) : Boolean
2421: function Sgn( const X : Extended) : Integer
2422: function SHA1(const fileName: string): string;
2423: function SHA256(astr: string; amode: char): string)
2424: function SHA512(astr: string; amode: char): string)
2425: function ShareMemoryManager : Boolean
2426: function ShellExecute(hWnd:HWND;Operation,FileN,Parameters,Dir:string;ShowCmd:Integer):integer;stdcall;
2427: function ShellExecute2(hwnd: HWND; const FileName: string):integer; stdcall;
2428: function ShellExecute3(aFilename: string; aParameters: string; aCommand:TS_ShellExecuteCmd): string;
2429: function SHORTCUT( KEY : WORD; SHIFT : TSHIFTSTATE) : TSHORTCUT
2430: function SHORTCUTTOTEXT( SHORTCUT : TSHORTCUT) : String
2431: function ShortDateFormat: string;
2432: function ShortenString(const DC:HDC;const S:WideString;const Width:Int;const
RTL:Bool;EllipsisWidth:Int):WideString
2433: function ShortTimeFormat: string;
2434: function SHOWMODAL: INTEGER
2435: function ShowWindow(C1: HWND; C2: integer): boolean;
2436: procedure ShowMemory //in Dialog
2437: function ShowMemory2: string;
2438: function ShutDownOS : Boolean

```



```

2439: Function Signe( const X, Y : Extended) : Extended
2440: Function Sign( const X : Extended) : Integer
2441: Function Sin(e : Extended) : Extended;
2442: Function sinc( const x : Double) : Double
2443: Function SinJ( X : Float) : Float
2444: Function Size( const AFileName : String) : Integer
2445: function SizeOf: Longint;
2446: Function SizeofResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : Integer
2447: function SlashSep(const Path, S: String): String
2448: Function SLNDepreciation( const Cost, Salvage : Extended; Life : Integer) : Extended
2449: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
2450: Function SmallPoint(X, Y: Integer): TSmallPoint)
2451: Function Soundex( const AText : string; ALength : TSoundexLength) : string
2452: Function SoundexCompare( const AText, AOther : string; ALength : TSoundexLength) : Integer
2453: Function SoundexInt( const AText : string; ALength : TSoundexIntLength) : Integer
2454: Function SoundexProc( const AText, AOther : string) : Boolean
2455: Function SoundexSimilar( const AText, AOther : string; ALength : TSoundexLength) : Boolean
2456: Function SoundexWord( const AText : string) : Word
2457: Function SourcePos : Longint
2458: function SourcePos:LongInt
2459: Function Split0( Str : string; const substr : string) : TStringList
2460: Procedure SplitNameValue( const Line : string; var Name, Value : string)
2461: Function SQLRequiresParams( const SQL : WideString) : Boolean
2462: Function Sqr(e : Extended) : Extended;
2463: Function Sqrt(e : Extended) : Extended;
2464: Function StartIP : String
2465: Function StartPan( WndHandle : THandle; AControl : TControl) : Boolean
2466: Function StartOfDay( const AYear, AMonth, ADay : Word) : TDateTime;
2467: Function StartOfDay1( const AYear, ADayOfYear : Word) : TDateTime;
2468: Function StartOfAMonth( const AYear, AMonth : Word) : TDateTime
2469: Function StartOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
2470: Function StartOfAYear( const AYear : Word) : TDateTime
2471: Function StartOfTheDay( const AValue : TDateTime) : TDateTime
2472: Function StartOfTheMonth( const AValue : TDateTime) : TDateTime
2473: Function StartOfTheWeek( const AValue : TDateTime) : TDateTime
2474: Function StartOfTheYear( const AValue : TDateTime) : TDateTime
2475: Function StartsStr( const ASubText, AText : string) : Boolean
2476: Function StartsText( const ASubText, AText : string) : Boolean
2477: Function StartsWith( const ANSIStr, APattern : String) : Boolean
2478: Function StartsWith( const str : string; const sub : string) : Boolean
2479: Function StartsWithACE( const ABytes : TIdBytes) : Boolean
2480: Function StatusString( StatusCode : Integer) : string
2481: Function StdDev( const Data : array of Double) : Extended
2482: Function Stop : Float
2483: Function StopCount( var Counter : TJclCounter) : Float
2484: Function StoreColumns : Boolean
2485: Function StrAfter( const sString : string; const sDelimiters : string) : string;
2486: Function StrAfter1( const sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2487: Function StrAlloc( Size : Cardinal) : PChar
2488: function StrAlloc(Size: Cardinal): PChar)
2489: Function StrBefore( const sString : string; const sDelimiters : string) : string;
2490: Function StrBefore1( const sString : string; const sDelimiters:string; out cDelimiter:char): string;
2491: Function StrBufSize( Str : PChar) : Cardinal
2492: function StrBufSize(const Str: PChar): Cardinal)
2493: Function StrByteType( Str : PChar; Index : Cardinal) : TmbcsByteType
2494: function StrByteType(Str: PChar; Index: Cardinal): TmbcsByteType)
2495: Function StrCat( Dest : PChar; Source : PChar) : PChar
2496: function StrCat(Dest: PChar; const Source: PChar): PChar)
2497: Function StrCharLength( Str : PChar) : Integer
2498: Function StrComp( Str1, Str2 : PChar) : Integer
2499: function StrComp(const Str1: PChar; const Str2: PChar): Integer)
2500: Function StrCopy( Dest : PChar; Source : PChar) : PChar
2501: function StrCopy(Dest: PChar; const Source: PChar): PChar)
2502: Function Stream_to_AnsiString( Source : TStream) : ansistring
2503: Function Stream_to_Base64( Source : TStream) : ansistring
2504: Function Stream_to_decimalbytes( Source : TStream) : string
2505: Function Stream2WideString( oStream : TStream) : WideString
2506: Function StreamtoAnsiString( Source : TStream) : ansistring
2507: Function StreamToByte( Source : TStream) : string
2508: Function StreamToDecimalbytes( Source : TStream) : string
2509: Function StreamtoOrd( Source : TStream) : string
2510: Function StreamToString( Source : TStream) : string
2511: Function StrECopy( Dest : PChar; Source : PChar) : PChar
2512: Function StrEmpty( const sString : string) : boolean
2513: Function StrEnd( Str : PChar) : PChar
2514: function StrEnd(const Str: PChar): PChar)
2515: Function StrFilter( const sString : string; xValidChars : TCharSet) : string
2516: Function StrFmt(Buffer, Format: PChar; const Args: array of const): PChar)
2517: Function StrGet(var S : String; I : Integer) : Char;
2518: Function StrGet2(S : String; I : Integer) : Char;
2519: Function StrHasPrefix( const sString : string; const sPrefix : string) : boolean
2520: Function StrHasSuffix( const sString : string; const sSuffix : string) : boolean
2521: Function StrHtmlDecode( const AStr : String) : String
2522: Function StrHtmlEncode( const AStr : String) : String
2523: Function StrToBytes(const Value: String): TBytes;
2524: Function StrIComp( Str1, Str2 : PChar) : Integer
2525: Function StringOfChar(c : char;I : longInt) : String;
2526: Function StringOfChar2( ch : WideChar; Count : Integer) : WideString;
2527: Function StringPad(InputStr,FillChar: String; StrLen:Integer; StrJustify:Boolean): String;

```

```

2528: Function StringRefCount(const s: String): integer;
2529: Function StringReplace( S, OldPattern, NewPattern: string; Flags: TReplaceFlags): string
2530: Function JStringReplace( const S, OldPattern, NewPattern: string; Flags: TReplaceFlags): string
2531: Function StringReplace(const SourceString, OldPattern, NewPattern: string; Flags: TReplaceFlags): string;
2532: Function StringRemove( const S, Pattern: string; Flags: TReplaceFlags): string
2533: Function StringToBoolean( const Ps: string): Boolean
2534: function StringToColor(const S: string): TColor
2535: function StringToCursor(const S: string): TCursor;
2536: function StringToGUID(const S: string): TGUID
2537: Function StringTokenizer( const str: string; const delim: string): IStringTokenizer
2538: Function StringToArray( const str: string; const delim: string): TStringDynArray
2539: Function StringWidth( S: string): Integer
2540: Function StrInternetToDateTime( Value: string): TDateTime
2541: Function StrIsDateTime( const Ps: string): Boolean
2542: Function StrIsFloatMoney( const Ps: string): Boolean
2543: Function StrIsInteger( const S: string): Boolean
2544: Function StrLCat( Dest: PChar; Source: PChar; MaxLen: Cardinal): PChar
2545: Function StrLComp( Str1, Str2: PChar; MaxLen: Cardinal): Integer
2546: Function StrLCopy( Dest: PChar; Source: PChar; MaxLen: Cardinal): PChar
2547: Function StrLen( Str: PChar): Cardinal
2548: function StrLen(const Str: PChar): Cardinal
2549: Function StrLessPrefix( const sString: string; const sPrefix: string): string
2550: Function StrLessSuffix( const sString: string; const sSuffix: string): string
2551: Function StrLIComp( Str1, Str2: PChar; MaxLen: Cardinal): Integer
2552: Function StrLower( Str: PChar): PChar
2553: Function StrMove( Dest: PChar; Source: PChar; Count: Cardinal): PChar
2554: function StrMove(Dest: PChar; const Source: PChar; Count: Cardinal): PChar
2555: Function StrNew( Str: PChar): PChar
2556: function StrNew(const Str: PChar): PChar
2557: Function StrNextChar( Str: PChar): PChar
2558: Function StrPad( const sString: string; const sPad: string; const iLength: integer): string
2559: Function StrParse( var sString: string; const sDelimiters: string): string;
2560: Function StrParse( var sString: string; const sDelimiters: string; out cDelimiter: char): string;
2561: Function StrPas( Str: PChar): string
2562: function StrPas(const Str: PChar): string
2563: Function StrPCopy( Dest: PChar; Source: string): PChar
2564: function StrPCopy(Dest: PChar; const Source: string): PChar
2565: Function StrPLCopy( Dest: PChar; Source: string; MaxLen: Cardinal): PChar
2566: Function StrPos( Str1, Str2: PChar): PChar
2567: Function StrScan(const Str: PChar; Chr: Char): PChar
2568: Function StrRScan(const Str: PChar; Chr: Char): PChar
2569: Function StrToBcd( const AValue: string): TBcd
2570: Function StrToBool( S: string): Boolean
2571: Function StrToBoolDef( S: string; Default: Boolean): Boolean
2572: Function StrToCard( const AStr: String): Cardinal
2573: Function StrToConv( AText: string; out AType: TConvType): Double
2574: Function StrToCurr( S: string): Currency;
2575: function StrToCurr(const S: string): Currency
2576: Function StrToCurrDef( S: string; Default: Currency): Currency;
2577: Function StrToDate( S: string): TDateTime;
2578: function StrToDate(const s: string): TDateTime;
2579: Function StrToDateDef( S: string; Default: TDateTime): TDateTime;
2580: Function StrToDateTime( S: string): TDateTime;
2581: function StrToDateTime(const S: string): TDateTime
2582: Function StrToDateTimeDef( S: string; Default: TDateTime): TDateTime;
2583: Function StrToDay( const ADay: string): Byte
2584: Function StrToFloat( S: string): Extended;
2585: function StrToFloat(s: String): Extended;
2586: Function StrToFloatDef( S: string; Default: Extended): Extended;
2587: function StrToFloatDef(const S: string; const Default: Extended): Extended
2588: Function StrToFloat( S: string): Extended;
2589: Function StrToFloat2( S: string; FormatSettings: TFormatSettings): Extended;
2590: Function StrToFloatDef( S: string; Default: Extended): Extended;
2591: Function StrToFloatDef2(S: string; Default: Extended; FormatSettings: TFormatSettings): Extended;
2592: Function StrToCurr( S: string): Currency;
2593: Function StrToCurr2( S: string; FormatSettings: TFormatSettings): Currency;
2594: Function StrToCurrDef( S: string; Default: Currency): Currency;
2595: Function StrToCurrDef2( S: string; Default: Currency; FormatSettings: TFormatSettings): Currency;
2596: Function StrToTime2( S: string; FormatSettings: TFormatSettings): TDateTime;
2597: Function StrToTimeDef( S: string; Default: TDateTime): TDateTime;
2598: Function StrToTimeDef2( S: string; Default: TDateTime; FormatSettings: TFormatSettings): TDateTime;
2599: Function TryStrToTime( S: string; Value: TDateTime): Boolean;
2600: Function StrToDateTime( S: string): TDateTime;
2601: Function StrToDateTime2( S: string; FormatSettings: TFormatSettings): TDateTime;
2602: Function StrToDateTimeDef( S: string; Default: TDateTime): TDateTime;
2603: Function StrToFloatRegionalIndependent(aValue: String; aDecimalSymbol: Char; aDigitGroupSymbol: Char): Extended
2604: Function StrToInt( S: string): Integer
2605: function StrToInt(s: String): Longint;
2606: Function StrToInt64( S: string): Int64
2607: function StrToInt64(s: String): int64;
2608: Function StrToInt64Def( S: string; Default: Int64): Int64
2609: function StrToInt64Def(const S: string; const Default: Int64): Int64
2610: Function StrToIntDef( S: string; Default: Integer): Integer
2611: function StrToIntDef(const S: string; Default: Integer): Integer
2612: function StrToIntDef(s: String; def: Longint): Longint;
2613: Function StrToMonth( const AMonth: string): Byte
2614: Function StrToTime( S: string): TDateTime;
2615: function StrToTime(const S: string): TDateTime
2616: Function StrToTimeDef( S: string; Default: TDateTime): TDateTime;

```

```

2617: Function StrToWorld( const Value : String ) : Word
2618: Function StrToXmlDate( const DateStr : string; const Format : string ) : string
2619: Function StrToXmlDateTime( const DateStr : string; const Format : string ) : string
2620: Function StrToXmlTime( const TimeStr : string; const Format : string ) : string
2621: Function StrUpper( Str : PChar ) : PChar
2622: Function StuffString( const AText : string; AStart, ALength : Cardinal; const ASubText : string ) : string
2623: Function Sum( const Data : array of Double ) : Extended
2624: Function SumFloatArray( const B : TDynFloatArray ) : Float
2625: Function SumInt( const Data : array of Integer ) : Integer
2626: Function SumOfSquares( const Data : array of Double ) : Extended
2627: Function SumPairProductFloatArray( const X, Y : TDynFloatArray ) : Float
2628: Function SumSquareDiffFloatArray( const B : TDynFloatArray; Diff : Float ) : Float
2629: Function SumSquareFloatArray( const B : TDynFloatArray ) : Float
2630: Function Supports( CursorOptions : TCursorOptions ) : Boolean
2631: Function SupportsClipboardFormat( AFormat : Word ) : Boolean
2632: Function SwapWord( w : word ) : word
2633: Function SwapInt( i : integer ) : integer
2634: Function SwapLong( L : longint ) : longint
2635: Function Swap( i : integer ) : integer
2636: Function SYDDepreciation( const Cost, Salvage : Extended; Life, Period : Integer ) : Extended
2637: Function SyncTime : Boolean
2638: Function SysErrorMessage( ErrorCode : Integer ) : string
2639: function SysErrorMessage( ErrorCode : Integer ) : string
2640: Function SystemTimeToDateTime( SystemTime : TSystemTime ) : TDateTime
2641: function SystemTimeToDateTime( const SystemTime : TSystemTime ) : TDateTime;
2642: Function SysStringLen( const S : WideString ) : Integer; stdcall;
2643: Function TabRect( Index : Integer ) : TRect
2644: Function Tan( const X : Extended ) : Extended
2645: Function TaskMessageDlg( const Title,
    Msg: string; DlgType: TMsgDlgType; Buttons: TMsgDlgButtons; HelpCtx: Longint ) : Integer;
2646: Function TaskMessageDlg1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
    HelpCtx : Longint; DefaultButton : TMsgDlgBtn ) : Integer;
2647: Function TaskMessageDlgPos( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
    HelpCtx : Longint; X, Y : Integer ) : Integer;
2648: Function TaskMessageDlgPos1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
    HelpCtx : Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn ) : Integer;
2649: Function TaskMessageDlgPosHelp( const Title, Msg : string; DlgType : TMsgDlgType; Buttons :
    TMsgDlgButtons; HelpCtx : Longint; X, Y : Integer; const HelpFileName : string ) : Integer;
2650: Function TaskMessageDlgPosHelp1( const Title, Msg: string; DlgType: TMsgDlgType; Buttons: TMsgDlgButtons;
    HelpCtx: Longint; X, Y: Integer; const HelpFileName: string; DefaultButton: TMsgDlgBtn ) : Integer;
2651: Function TenToY( const Y : Float ) : Float
2652: Function TerminateApp( ProcessID : DWORD; Timeout : Integer ) : TJclTerminateAppResult
2653: Function TerminateTask( Wnd : HWND; Timeout : Integer ) : TJclTerminateAppResult
2654: Function TestBit( const Value : Byte; const Bit : TBitRange ) : Boolean;
2655: Function TestBit2( const Value : Shortint; const Bit : TBitRange ) : Boolean;
2656: Function TestBit3( const Value : Smallint; const Bit : TBitRange ) : Boolean;
2657: Function TestBit4( const Value : Word; const Bit : TBitRange ) : Boolean;
2658: Function TestBit5( const Value : Cardinal; const Bit : TBitRange ) : Boolean;
2659: Function TestBit6( const Value : Integer; const Bit : TBitRange ) : Boolean;
2660: Function TestBit7( const Value : Int64; const Bit : TBitRange ) : Boolean;
2661: Function TestBits( const Value, Mask : Byte ) : Boolean;
2662: Function TestBits1( const Value, Mask : Shortint ) : Boolean;
2663: Function TestBits2( const Value, Mask : Smallint ) : Boolean;
2664: Function TestBits3( const Value, Mask : Word ) : Boolean;
2665: Function TestBits4( const Value, Mask : Cardinal ) : Boolean;
2666: Function TestBits5( const Value, Mask : Integer ) : Boolean;
2667: Function TestBits6( const Value, Mask : Int64 ) : Boolean;
2668: Function TestFDIVInstruction : Boolean
2669: Function TestStreamFormat( Stream : TStream ) : TStreamOriginalFormat
2670: Function TextExtent( const Text : string ) : TSize
2671: function TextHeight( Text : string ) : Integer;
2672: Function TextIsSame( const A1 : string; const A2 : string ) : Boolean
2673: Function TextStartsWith( const S, SubS : string ) : Boolean
2674: function TextToFloat( Buffer: PChar; var Value: Extended; ValueType: TFloatValue ) : Boolean
2675: Function ConvInteger( i : integer ) : string;
2676: Function IntegerToText( i : integer ) : string;
2677: Function TEXTTOSHORTCUT( TEXT : String ) : TSHORTCUT
2678: Function TextWidth( Text : string ) : Integer;
2679: Function ThreadCount : integer
2680: function ThousandSeparator: char;
2681: Function Ticks : Cardinal
2682: Function Time : TDateTime
2683: function Time: TDateTime;
2684: function TimeGetTime: int64;
2685: Function TimeOf( const AValue : TDateTime ) : TDateTime
2686: function TimeSeparator: char;
2687: function TimeStampToDateTime( const TimeStamp : TTimeStamp ) : TDateTime
2688: Function TimeStampToMsecs( TimeStamp : TTimeStamp ) : Comp
2689: function TimeStampToMsecs( const TimeStamp : TTimeStamp ) : Comp
2690: Function TimeToStr( DateTime : TDateTime ) : string;
2691: function TimeToStr( const DateTime : TDateTime ) : string;
2692: Function TimeZoneBias : TDateTime
2693: Function ToCommon( const AValue : Double ) : Double
2694: function ToCommon( const AValue : Double ) : Double;
2695: Function Today : TDateTime
2696: Function ToggleBit( const Value : Byte; const Bit : TBitRange ) : Byte;
2697: Function ToggleBit1( const Value : Shortint; const Bit : TBitRange ) : Shortint;
2698: Function ToggleBit2( const Value : Smallint; const Bit : TBitRange ) : Smallint;
2699: Function ToggleBit3( const Value : Word; const Bit : TBitRange ) : Word;

```

```

2700: Function ToggleBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2701: Function ToggleBit5( const Value : Integer; const Bit : TBitRange) : Integer;
2702: Function ToggleBit6( const Value : Int64; const Bit : TBitRange) : Int64;
2703: function TokenComponentIdent:String
2704: Function TokenFloat : Extended
2705: function TokenFloat:Extended
2706: Function TokenInt : Longint
2707: function TokenInt:LongInt
2708: Function TokenString : string
2709: function TokenString:String
2710: Function TokenSymbolIs( const S : string) : Boolean
2711: function TokenSymbolIs(S:String):Boolean
2712: Function Tomorrow : TDateTime
2713: Function ToRightOf( const pc : TControl; piSpace : Integer) : Integer
2714: Function ToString : string
2715: Function TotalVariance( const Data : array of Double) : Extended
2716: Function Trace2( AURL : string) : string;
2717: Function TrackMenu( Button : TToolButton) : Boolean
2718: Function TRANSLATE( SRC, DEST : PCHAR; TOEM : BOOLEAN) : INTEGER
2719: Function TranslateURI( const URI : string) : string
2720: Function TranslationMatchesLanguages( Exact : Boolean) : Boolean
2721: Function TransparentStretchBlt( DstDC : HDC; DstX, DstY, DstW, DstH: Integer; SrcDC:HDC; SrcX, SrcY, SrcW,
SrcH: Integer; MaskDC : HDC; MaskX, MaskY : Integer) : Boolean
2722: Function Trim( S : string) : string;
2723: Function Trim( S : WideString) : WideString;
2724: Function Trim(s : AnyString) : AnyString;
2725: Function TrimAllOf( ATrim, AText : String) : String
2726: Function TrimLeft( S : string) : string;
2727: Function TrimLeft( S : WideString) : WideString;
2728: function TrimLeft(const S: string): string)
2729: Function TrimRight( S : string) : string;
2730: Function TrimRight( S : WideString) : WideString;
2731: function TrimRight(const S: string): string)
2732: function TrueBoolStrs: array of string
2733: Function Trunc(e : Extended) : Longint;
2734: Function Trunc64(e: extended): Int64;
2735: Function TruncPower( const Base, Exponent : Float) : Float
2736: Function TryConvTypeToFamily( const AFrom, ATo : TConvType; out AFamily : TConvFamily) : Boolean;
2737: Function TryConvTypeToFamily1( const AType : TConvType; out AFamily : TConvFamily) : Boolean;
2738: function TryEncodeDate(Year, Month, Day: Word; var Date: TDateTime): Boolean;
2739: Function TryEncodeDateDay( const AYear, ADayOfYear : Word; out AValue : TDateTime) : Boolean
2740: Function TryEncodeDateMonthWeek(const AY, AMonth, AWeekOfMonth, ADayOfWeek: Word; var AValue: TDateTime): Boolean
2741: Function TryEncodeDateTime(const AYear, AMonth, ADay, AHour, AMin, ASec, AMilliSecond: Word; out
AValue: TDateTime): Boolean
2742: Function TryEncodeDateWeek(const AY, AWeekOfYear: Word; out AValue: TDateTime; const ADayOfWeek: Word): Boolean
2743: Function TryEncodeDayOfWeekInMonth(const AYear, AMonth, ANthDayOfWeek, ADayOfWeek: Word; out
AVal: TDateTime): Bool
2744: function TryEncodeTime(Hour, Min, Sec, MSec: Word; var Time: TDateTime): Boolean;
2745: Function TryFloatToDateTime( Value : Extended; AResult : TDateTime) : Boolean
2746: Function TryJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2747: Function TryLock : Boolean
2748: Function TryModifiedJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2749: Function TryRecodeDateTime( const AValue : TDateTime; const AYear, AMonth, ADay, AHour, AMinute, ASecond,
AMilliSecond : Word; out AResult : TDateTime) : Boolean
2750: Function TryStrToBcd( const AValue : string; var Bcd : TBcd) : Boolean
2751: Function TryStrToConv( AText : string; out AValue : Double; out AType : TConvType) : Boolean
2752: Function TryStrToDate( S : string; Value : TDateTime) : Boolean;
2753: Function TryStrToDateTime( S : string; Value : TDateTime) : Boolean;
2754: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2755: Function TryStrToInt(const S: AnsiString; var I: Integer): Boolean;
2756: Function TryStrToInt64(const S: AnsiString; var I: Int64): Boolean;
2757: Function TwoByteToWord( AByte1, AByte2 : Byte) : Word
2758: Function TwoCharToWord( AChar1, AChar2 : Char) : Word
2759: Function TwoToY( const Y : Float) : Float
2760: Function UCS4StringToWideString( const S : UCS4String) : WideString
2761: Function UIDL( const ADest : TStrings; const AMsgNum : Integer) : Boolean
2762: function Unassigned: Variant;
2763: Function UndoLastChange( FollowChange : Boolean) : Boolean
2764: function UniCodeToStr(Value: string): string;
2765: Function UnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
2766: function UnionRect(out Rect: TRect; const R1, R2: TRect): Boolean)
2767: Function UnixDateTimeToDelphiDateTime( UnixDateTime : Cardinal) : TDateTime
2768: Function UnixPathToDosPath( const Path : string) : string
2769: Function UnixToDateTime( const AValue : Int64) : TDateTime
2770: function UnixToDateTime(U: Int64): TDateTime;
2771: Function UnlockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HRESULT
2772: Function UnlockResource( ResData : HGLOBAL) : LongBool
2773: Function UnlockVolume( var Handle : THandle) : Boolean
2774: Function UnMaskString( Mask, Value : String) : String
2775: function UpCase(ch : Char) : Char;
2776: Function UpCaseFirst( const AStr : string) : string
2777: Function UpCaseFirstWord( const AStr : string) : string
2778: Function UpdateAction( Action : TBasicAction) : Boolean
2779: Function UpdateKind : TUpdateKind
2780: Function UPDATESTATUS : TUPDATESTATUS
2781: Function UpperCase( S : string) : string
2782: Function Uppercase(s : AnyString) : AnyString;
2783: Function URLEnCode( ASrc : string) : string
2784: Function URLEnCode( const ASrc : string) : string

```



```

2785: Function UseRightToLeftAlignment : Boolean
2786: Function UseRightToLeftAlignmentForField( const AField : TField; Alignment : TAlignment) : Boolean
2787: Function UseRightToLeftReading : Boolean
2788: Function UTF8CharLength( Lead : Char) : Integer
2789: Function UTF8CharSize( Lead : Char) : Integer
2790: Function UTF8Decode( const S : UTF8String) : WideString
2791: Function UTF8Encode( const WS : WideString) : UTF8String
2792: Function UTF8LowerCase( const S : UTF8string) : UTF8string
2793: Function Utf8ToAnsi( const S : UTF8String) : string
2794: Function Utf8ToAnsiEx( const S : UTF8String; const cp : integer) : string
2795: Function UTF8UpperCase( const S : UTF8string) : UTF8string
2796: Function ValidFieldIndex( FieldIndex : Integer) : Boolean
2797: Function ValidParentForm(control: TControl): TForm
2798: Function Value : Variant
2799: Function ValueExists( const Section, Ident : string) : Boolean
2800: Function ValueOf( const Key : string) : Integer
2801: Function ValueInSet(AValue: Variant; ASet: Variant): Boolean;
2802: Function VALUEOFKEY( const AKEY : VARIANT) : VARIANT
2803: Function VarArrayFromStrings( Strings : TStrings) : Variant
2804: Function VarArrayFromWideStrings( Strings : TWideStrings) : Variant
2805: Function VarArrayGet(var S : Variant; I : Integer) : Variant;
2806: Function VarFMTBcd : TVarType
2807: Function VarFMTBcdCreatel : Variant;
2808: Function VarFMTBcdCreate2( const AValue : string; Precision, Scale : Word) : Variant;
2809: Function VarFMTBcdCreate3( const AValue : Double; Precision : Word; Scale : Word) : Variant;
2810: Function VarFMTBcdCreate4( const ABcd : TBcd) : Variant;
2811: Function Variance( const Data : array of Double) : Extended
2812: Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
2813: Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
2814: Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
2815: Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
2816: Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
2817: Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
2818: Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
2819: Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
2820: Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
2821: Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
2822: Function VariantNeg( const V1 : Variant) : Variant
2823: Function VariantNot( const V1 : Variant) : Variant
2824: Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
2825: Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
2826: Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
2827: Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
2828: Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
2829: function VarIsEmpty(const V: Variant): Boolean;
2830: Function VarIsFMTBcd( const AValue : Variant) : Boolean;
2831: function VarIsNull(const V: Variant): Boolean;
2832: Function VarToBcd( const AValue : Variant) : TBcd
2833: function VarType(const V: Variant): TVarType;
2834: Function VarType( const V : Variant) : TVarType
2835: Function VarAsType( const V : Variant; AVarType : TVarType) : Variant
2836: Function VarIsType( const V : Variant; AVarType : TVarType) : Boolean;
2837: Function VarIsType1( const V : Variant; const AVarTypes : array of TVarType) : Boolean;
2838: Function VarIsByRef( const V : Variant) : Boolean
2839: Function VarIsEmpty( const V : Variant) : Boolean
2840: Procedure VarCheckEmpty( const V : Variant)
2841: Function VarIsNull( const V : Variant) : Boolean
2842: Function VarIsClear( const V : Variant) : Boolean
2843: Function VarIsCustom( const V : Variant) : Boolean
2844: Function VarIsOrdinal( const V : Variant) : Boolean
2845: Function VarIsFloat( const V : Variant) : Boolean
2846: Function VarIsNumeric( const V : Variant) : Boolean
2847: Function VarIsStr( const V : Variant) : Boolean
2848: Function VarToStr( const V : Variant) : string
2849: Function VarToStrDef( const V : Variant; const ADefault : string) : string
2850: Function VarToWideStr( const V : Variant) : WideString
2851: Function VarToWideStrDef( const V : Variant; const ADefault : WideString) : WideString
2852: Function VarToDateTime( const V : Variant) : TDateTime
2853: Function VarFromDateTime( const DateTime : TDateTime) : Variant
2854: Function VarInRange( const AValue, AMin, AMax : Variant) : Boolean
2855: Function VarEnsureRange( const AValue, AMin, AMax : Variant) : Variant
2856: TVariantRelationship, '( ( vrEqual, vrLessThan, vrGreaterThan, vrNotEqual )
2857: Function VarSameValue( const A, B : Variant) : Boolean
2858: Function VarCompareValue( const A, B : Variant) : TVariantRelationship
2859: Function VarIsEmptyParam( const V : Variant) : Boolean
2860: Function VarIsError( const V : Variant; out AResult : HRESULT) : Boolean;
2861: Function VarIsError1( const V : Variant) : Boolean;
2862: Function VarAsError( AResult : HRESULT) : Variant
2863: Procedure VarCopyNoInd( var Dest : Variant; const Source : Variant)
2864: Function VarIsArray( const A : Variant) : Boolean;
2865: Function VarIsArray1( const A : Variant; AResolveByRef : Boolean) : Boolean;
2866: Function VarArrayCreate( const Bounds : array of Integer; AVarType : TVarType) : Variant
2867: Function VarArrayOf( const Values : array of Variant) : Variant
2868: Function VarArrayRef( const A : Variant) : Variant
2869: Function VarTypeIsValidArrayType( const AVarType : TVarType) : Boolean
2870: Function VarTypeIsValidElementType( const AVarType : TVarType) : Boolean
2871: Function VarArrayDimCount( const A : Variant) : Integer
2872: Function VarArrayLowBound( const A : Variant; Dim : Integer) : Integer
2873: Function VarArrayHighBound( const A : Variant; Dim : Integer) : Integer

```

```

2874: Function VarArrayLock( const A : Variant ) : ___Pointer
2875: Procedure VarArrayUnlock( const A : Variant )
2876: Function VarArrayGet( const A : Variant; const Indices : array of Integer ) : Variant
2877: Procedure VarArrayPut( var A : Variant; const Value : Variant; const Indices : array of Integer )
2878: Procedure DynArrayToVariant( var V : Variant; const DynArray : ___Pointer; TypeInfo : ___Pointer )
2879: Procedure DynArrayFromVariant( var DynArray : ___Pointer; const V : Variant; TypeInfo : ___Pointer )
2880: Function Unassigned : Variant
2881: Function Null : Variant
2882: Function VectorAdd( const V1, V2 : TFloatPoint ) : TFloatPoint
2883: function VectorAdd(const V1,V2: TFloatPoint): TFloatPoint;
2884: Function VectorDot( const V1, V2 : TFloatPoint ) : Double
2885: function VectorDot(const V1,V2: TFloatPoint): Double;
2886: Function VectorLengthSqr( const V : TFloatPoint ) : Double
2887: function VectorLengthSqr(const V: TFloatPoint): Double;
2888: Function VectorMult( const V : TFloatPoint; const s : Double ) : TFloatPoint
2889: function VectorMult(const V: TFloatPoint; const s: Double): TFloatPoint;
2890: Function VectorSubtract( const V1, V2 : TFloatPoint ) : TFloatPoint
2891: function VectorSubtract(const V1,V2: TFloatPoint): TFloatPoint;
2892: Function Verify( AUserName : String ) : String
2893: Function Versine( X : Float ) : Float
2894: function VersionCheck: boolean;
2895: Function VersionLanguageId( const LangIdRec : TLangIdRec ) : string
2896: Function VersionLanguageName( const LangId : Word ) : string
2897: Function VersionResourceAvailable( const FileName : string ) : Boolean
2898: Function Visible : Boolean
2899: function VolumeID(DriveChar: Char): string
2900: Function WaitFor( const AString : string ) : string
2901: Function WaitFor( const TimeOut : Cardinal ) : TWaitResult
2902: Function WaitFor1 : TWaitResult;
2903: Function WaitForData( Timeout : Longint ) : Boolean
2904: Function WebColorNameToColor( WebColorName : string ) : TColor
2905: Function WebColorStrToColor( WebColor : string ) : TColor
2906: Function WebColorToRGB( WebColor : Integer ) : Integer
2907: Function wGet(aURL, afile: string): boolean;
2908: Function wGet2(aURL, afile: string): boolean; //without file open
2909: Function WebGet(aURL, afile: string): boolean;
2910: Function WebExists: boolean; //alias to isinternet
2911: Function WeekOf( const AValue : TDateTime ) : Word
2912: Function WeekOfTheMonth( const AValue : TDateTime ) : Word;
2913: Function WeekOfTheMonth1( const AValue : TDateTime; var AYear, AMonth : Word ) : Word;
2914: Function WeekOfTheYear( const AValue : TDateTime ) : Word;
2915: Function WeekOfTheYear1( const AValue : TDateTime; var AYear : Word ) : Word;
2916: Function WeeksBetween( const ANow, AThen : TDateTime ) : Integer
2917: Function WeeksInAYear( const AYear : Word ) : Word
2918: Function WeeksInYear( const AValue : TDateTime ) : Word
2919: Function WeekSpan( const ANow, AThen : TDateTime ) : Double
2920: Function WideAdjustLineBreaks( const S : WideString; Style : TTextLineBreakStyle ) : WideString
2921: Function WideCat( const x, y : WideString ) : WideString
2922: Function WideCompareStr( S1, S2 : WideString ) : Integer
2923: function WideCompareStr(const S1: WideString; const S2: WideString): Integer
2924: Function WideCompareText( S1, S2 : WideString ) : Integer
2925: function WideCompareText(const S1: WideString; const S2: WideString): Integer
2926: Function WideCopy( const src : WideString; index, count : Integer ) : WideString
2927: Function WideDequotedStr( const S : WideString; AQuote : WideChar ) : WideString
2928: Function WideEqual( const x, y : WideString ) : Boolean
2929: function WideFormat(const Format: WideString; const Args: array of const): WideString)
2930: Function WideGreater( const x, y : WideString ) : Boolean
2931: Function WideLength( const src : WideString ) : Integer
2932: Function WideLess( const x, y : WideString ) : Boolean
2933: Function WideLowerCase( S : WideString ) : WideString
2934: function WideLowerCase(const S: WideString): WideString
2935: Function WidePos( const src, sub : WideString ) : Integer
2936: Function WideQuotedStr( const S : WideString; Quote : WideChar ) : WideString
2937: Function WideReplaceStr( const AText, AFromText, AToText : WideString ) : WideString
2938: Function WideReplaceText( const AText, AFromText, AToText : WideString ) : WideString
2939: Function WideSameStr( S1, S2 : WideString ) : Boolean
2940: function WideSameStr(const S1: WideString; const S2: WideString): Boolean
2941: Function WideSameText( S1, S2 : WideString ) : Boolean
2942: function WideSameText(const S1: WideString; const S2: WideString): Boolean
2943: Function WideStringReplace( const S, OldPattern, NewPattern : WideString; Flags: TReplaceFlags ) : WideString
2944: Function WideStringToUCS4String( const S : WideString ) : UCS4String
2945: Function WideUpperCase( S : WideString ) : WideString
2946: Function Win32BackupFile( const FileName : string; Move : Boolean ) : Boolean
2947: function Win32Check(RetVal: boolean): boolean
2948: Function Win32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean ) : Boolean
2949: Function Win32RestoreFile( const FileName : string ) : Boolean
2950: Function Win32Type : TIdWin32Type
2951: Function WinColor( const Color32 : TColor32 ) : TColor
2952: function winexec(FileName: pchar; showCmd: integer): integer;
2953: Function WinExec32( const Cmd : string; const CmdShow : Integer ) : Boolean
2954: Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer ) : Cardinal
2955: Function WithinPastDays( const ANow, AThen : TDateTime; const ADays : Integer ) : Boolean
2956: Function WithinPastHours( const ANow, AThen : TDateTime; const AHours : Int64 ) : Boolean
2957: Function WithinPastMilliseconds( const ANow, AThen : TDateTime; const AMilliSeconds : Int64 ) : Boolean
2958: Function WithinPastMinutes( const ANow, AThen : TDateTime; const AMinutes : Int64 ) : Boolean
2959: Function WithinPastMonths( const ANow, AThen : TDateTime; const AMonths : Integer ) : Boolean
2960: Function WithinPastSeconds( const ANow, AThen : TDateTime; const ASeconds : Int64 ) : Boolean
2961: Function WithinPastWeeks( const ANow, AThen : TDateTime; const AWeeks : Integer ) : Boolean
2962: Function WithinPastYears( const ANow, AThen : TDateTime; const AYears : Integer ) : Boolean

```

```

2963: Function WNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PChar ) : DWORD
2964: Function WordToStr( const Value : Word ) : String
2965: Function WordGridFormatIdentToInt( const Ident : string; var Value : Longint ) : Boolean
2966: Function IntToWorldGridFormatIdent( Value : Longint; var Ident : string ) : Boolean
2967: Procedure GetWordGridFormatValues( Proc : TGetStrProc)
2968: Function WorkArea : Integer
2969: Function WrapText( Line : string; MaxCol : Integer ) : string;
2970: Function WrapText( Line, BreakStr : string; BreakChars : TSysCharSet; MaxCol : Integer ) : string;
2971: Function Write( pv : Pointer; cb : Longint; pcbWritten : PLongint ) : HRESULT
2972: function Write(Buffer:String;Count:LongInt):LongInt
2973: Function WriteClient( var Buffer, Count : Integer ) : Integer
2974: Function WriteFile( const AFile : string; const AEnableTransferFile : Boolean ) : Cardinal
2975: Function WriteHeaders( StatusCode : Integer; const ReasonString, Headers : string ) : Boolean
2976: Function WriteString( const AString : string ) : Boolean
2977: Function WStrGet(var S : AnyString; I : Integer ) : WideChar;
2978: Function wvsprintf( Output : PChar; Format : PChar; arglist : va_list ) : Integer
2979: Function wsprintf( Output : PChar; Format : PChar ) : Integer
2980: Function XmlDateTimeToStr( const XmlDateTime : string; const Format : string ) : string
2981: Function XmlTimeToStr( const XmlTime : string; const Format : string ) : string
2982: Function XorDecode( const Key, Source : string ) : string
2983: Function XorEncode( const Key, Source : string ) : string
2984: Function XorString( const Key, Src : ShortString ) : ShortString
2985: Function Yield : Bool
2986: Function YearOf( const AValue : TDateTime ) : Word
2987: Function YearsBetween( const ANow, AThen : TDateTime ) : Integer
2988: Function YearSpan( const ANow, AThen : TDateTime ) : Double
2989: Function Yesterday : TDateTime
2990: Function YesNoDialog(const ACaption, AMsg: string): boolean;
2991: Function ( const Name : string; Proc : TUserFunction)
2992: Function using Special_Scholz from 3.8.5.0
2993: Function TimeToFloat(value:Extended):Extended; // Normalstunden --> Industriestunden
2994: Function FloatToTime(value:Extended):Extended; // Industriestunden --> Normalstunden
2995: Function FloatToTime2Dec(value:Extended):Extended;
2996: Function MinToStd(value:Extended):Extended;
2997: Function MinToStdAsString(value:Extended):String;
2998: Function RoundFloatToStr(zahl:Extended; decimals:integer):String;
2999: Function RoundFloat(zahl:Extended; decimals:integer):Extended;
3000: Function Round2Dec (zahl:Extended):Extended;
3001: Function GetAngle(x,y:Extended):Double;
3002: Function AddAngle(a1,a2:Double):Double;
3003:
3004: *****
3005: unit uPSI_StText;
3006: *****
3007: Function TextSeek( var F : TextFile; Target : LongInt ) : Boolean
3008: Function TextFileSize( var F : TextFile ) : LongInt
3009: Function TextPos( var F : TextFile ) : LongInt
3010: Function TextFlush( var F : TextFile ) : Boolean
3011:
3012: *****
3013: from JvVCLUtils;
3014: *****
3015: { Windows resources (bitmaps and icons) VCL-oriented routines }
3016: procedure DrawBitmapTransparent(Dest:TCanvas;DstX,DstY:Integer;Bitmap:TBitmap;TransparentColor:TColor);
3017: procedure DrawBitmapRectTransparent(Dest: TCanvas;DstX,
  DstY:Integer;SrcRect:TRect;Bitmap:TBitmap;TransparColor:TColor);
3018: procedure StretchBitmapRectTransparent(Dest: TCanvas; DstX, DstY, DstW,DstH: Integer; SrcRect: TRect;
  Bitmap: TBitmap; TransparentColor: TColor);
3019: function MakeBitmap(ResID: PChar): TBitmap;
3020: function MakeBitmapID(ResID: Word): TBitmap;
3021: function MakeModuleBitmap(Module: THandle; ResID: PChar): TBitmap;
3022: function CreateTwoColorsBrushPattern(Color1, Color2: TColor): TBitmap;
3023: function CreateDisabledBitmap_NewStyle(FOriginal: TBitmap; BackColor: TColor): TBitmap;
3024: function CreateDisabledBitmapEx(FOriginal: TBitmap; OutlineColor, BackColor,
  HighlightColor, ShadowColor: TColor; DrawHighlight: Boolean): TBitmap;
3025: function CreateDisabledBitmap(FOriginal: TBitmap; OutlineColor: TColor): TBitmap;
3026: function ChangeBitmapColor(Bitmap: TBitmap; Color, NewColor: TColor): TBitmap;
3027: function AssignBitmapCell(Source: TGraphic; Dest: TBitmap; Cols, Rows, Index: Integer);
3028: procedure AssignBitmapCell(Source: TGraphic; Dest: TBitmap; Cols, Rows, Index: Integer);
3029: {$IFDEF WIN32}
3030: procedure ImageListDrawDisabled(Images: TImageList; Canvas: TCanvas;
  X, Y, Index: Integer; HighlightColor, GrayColor: TColor; DrawHighlight: Boolean);
3031: X, Y, Index: Integer; HighlightColor, GrayColor: TColor; DrawHighlight: Boolean);
3032: {$ENDIF}
3033: function MakeIcon(ResID: PChar): TIcon;
3034: function MakeIconID(ResID: Word): TIcon;
3035: function MakeModuleIcon(Module: THandle; ResID: PChar): TIcon;
3036: function CreateBitmapFromIcon(Icon: TIcon; BackColor: TColor): TBitmap;
3037: {$IFDEF WIN32}
3038: function CreateIconFromBitmap(Bitmap: TBitmap; TransparentColor: TColor): TIcon;
3039: {$ENDIF}
3040: { Service routines }
3041: procedure NotImplemented;
3042: procedure ResourceNotFound(ResID: PChar);
3043: function PointInRect(const P: TPoint; const R: TRect): Boolean;
3044: function PointInPolyRgn(const P: TPoint; const Points: array of TPoint): Boolean;
3045: function PaletteColor(Color: TColor): Longint;
3046: function WidthOf(R: TRect): Integer;
3047: function HeightOf(R: TRect): Integer;
3048: procedure PaintInverseRect(const RectOrg, RectEnd: TPoint);
3049: procedure DrawInvertFrame(ScreenRect: TRect; Width: Integer);

```

```

3050: procedure CopyParentImage(Control: TControl; Dest: TCanvas);
3051: procedure Delay(MSecs: Longint);
3052: procedure CenterControl(Control: TControl);
3053: Function PaletteEntries( Palette : HPALETTE ) : Integer
3054: Function WindowClassName( Wnd : HWND ) : string
3055: Function ScreenWorkArea : TRect
3056: Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3057: Procedure SwitchToWindow( Wnd : HWND; Restore : Boolean)
3058: Procedure ActivateWindow( Wnd : HWND)
3059: Procedure ShowWinNoAnimate( Handle : HWND; CmdShow : Integer)
3060: Procedure CenterWindow( Wnd : HWND)
3061: Procedure ShadeRect( DC : HDC; const Rect : TRect)
3062: Procedure KillMessage( Wnd : HWND; Msg : Cardinal)
3063: Function DialogsToPixelsX( Dlgs : Word ) : Word
3064: Function DialogsToPixelsY( Dlgs : Word ) : Word
3065: Function PixelsToDialogsX( Pixs : Word ) : Word
3066: Function PixelsToDialogsY( Pixs : Word ) : Word
3067: {$IFDEF WIN32}
3068: procedure ShowMDIClientEdge(ClientHandle: THandle; ShowEdge: Boolean);
3069: function MakeVariant(const Values: array of Variant): Variant;
3070: {$ENDIF}
3071: function CreateRotatedFont(Font: TFont; Angle: Integer): HFONT;
3072: function MsgBox(const Caption, Text: string; Flags: Integer): Integer;
3073: function MsgDlg(const Msg: string; AType: TMsgDlgType; AButtons: TMsgDlgButtons; HelpCtx: Longint): Word;
3074: {$IFDEF CBUILDER}
3075: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): HWND;
3076: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
3077: {$ELSE}
3078: function FindPrevInstance(const MainFormClass, ATitle: string): HWND;
3079: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
3080: {$ENDIF CBUILDER}
3081: function IsForegroundTask: Boolean;
3082: procedure MergeForm(AControl: TWinControl; AForm: TForm; Align: TAlign; Show: Boolean);
3083: function GetAveCharSize(Canvas: TCanvas): TPoint;
3084: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
3085: procedure FreeUnusedOle;
3086: procedure Beep;
3087: function GetWindowsVersionJ: string;
3088: function LoadDLL(const LibName: string): THandle;
3089: function RegisterServer(const ModuleName: string): Boolean;
3090: {$IFDEF WIN32}
3091: function IsLibrary: Boolean;
3092: {$ENDIF}
3093: { Gradient filling routine }
3094: type TFillDirection = (fdTopToBottom, fdBottomToTop, fdLeftToRight, fdRightToLeft);
3095: procedure GradientFillRect(Canvas: TCanvas; ARect: TRect; StartColor, EndColor: TColor; Direction:
TFillDirection; Colors: Byte);
3096: { String routines }
3097: function GetEnvVar(const VarName: string): string;
3098: function AnsiUpperFirstChar(const S: string): string;
3099: function StringToPChar(var S: string): PChar;
3100: function StrPAlloc(const S: string): PChar;
3101: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
3102: function DropT(const S: string): string;
3103: { Memory routines }
3104: function AllocMemo(Size: Longint): Pointer;
3105: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
3106: procedure FreeMemo(var fpBlock: Pointer);
3107: function GetMemoSize(fpBlock: Pointer): Longint;
3108: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
3109: {$IFDEF COMPILER5_UP}
3110: procedure FreeAndNil(var Obj);
3111: {$ENDIF}
3112: // from PNGLoader
3113: Function OptimizeForPNG(Image:TLinearBitmap;QuantizationSteps:Integer;TransparentColor:TColor):Integer
3114: Procedure TransformRGB2LOCO( Image : TLinearBitmap)
3115: Procedure TransformLOCO2RGB( Image : TLinearBitmap)
3116: Procedure SortPalette( const Pal : TPalette; var ColorMap : TColorMap)
3117: Function DrawButtonFace( Canvas : TCanvas; const Client : TRect; BevelWidth : Integer; Style :
TButtonStyle; IsRounded, IsDown, IsFocused : Boolean) : TRect //TButtons
3118: Function IsAnAllResult( const AModalResult : TModalResult ) : Boolean
3119: Function InitWndProc( HWindow : HWND; Message, WParam : Longint; LParam : Longint ) : Longint
3120: AddConstantN('CTL3D_ALL','LongWord').SetUInt( $FFFF);
3121: //Procedure ChangeBiDiModeAlignment( var Alignment : TAlignment)
3122: //Function SendAppMessage( Msg : Cardinal; WParam, LParam : Longint ) : Longint
3123: //Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3124: Procedure SetImeMode( hWnd : HWND; Mode : TImeMode)
3125: Procedure SetImeName( Name : TImeName)
3126: Function Win32NLSEnableIME( hWnd : HWND; Enable : Boolean ) : Boolean
3127: Function Imm32GetContext( hWnd : HWND ) : HIMC
3128: Function Imm32ReleaseContext( hWnd : HWND; hImc : HIMC ) : Boolean
3129: Function Imm32GetConversionStatus( hImc : HIMC; var Conversion, Sentence : longword ) : Boolean
3130: Function Imm32SetConversionStatus( hImc : HIMC; Conversion, Sentence : longword ) : Boolean
3131: Function Imm32SetOpenStatus( hImc : HIMC; fOpen : Boolean ) : Boolean
3132: // Function Imm32SetCompositionWindow( hImc : HIMC; lpCompForm : PCOMPOSITIONFORM ) : Boolean
3133: //Function Imm32SetCompositionFont( hImc : HIMC; lpLogFont : PLOGFONTA ) : Boolean
3134: Function Imm32GetCompositionString(hImc:HIMC;dWord1:longword;lpBuf:string;dwBufLen:longint):Longint
3135: Function Imm32IsIME( hK1 : longword ) : Boolean
3136: Function Imm32NotifyIME( hImc : HIMC; dwAction, dwIndex, dwValue:longword):Boolean

```



```

3137: Procedure DragDone( Drop : Boolean)
3138:
3139:
3140: *****added from jvjvclutils
3141: function CanvasMaxTextHeight(Canvas: TCanvas): Integer;
3142: function ReplaceComponentReference(This,NewReference:TComponent;var VarReference:TComponent):Boolean;
3143: procedure DrawLine(Canvas: TCanvas; X, Y, X2, Y2: Integer);
3144: function IsPositiveResult(Value: TModalResult): Boolean;
3145: function IsNegativeResult(Value: TModalResult): Boolean;
3146: function IsAbortResult(const Value: TModalResult): Boolean;
3147: function StripAllFromResult(const Value: TModalResult): TModalResult;
3148: // returns either BrightColor or DarkColor depending on the luminance of AColor
3149: // This function gives the same result (AFAIK) as the function used in Windows to
3150: // calculate the desktop icon text color based on the desktop background color
3151: function SelectColorByLuminance(AColor, DarkColor, BrightColor: TColor): TColor;
3152: type TJvHTMLCalcType = (htmlShow, htmlCalcWidth, htmlCalcHeight, htmlHyperLink);
3153:
3154: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3155:   const State: TOwnerDrawState; const Text: string; var Width: Integer;
3156:   CalcType: TJvHTMLCalcType; MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3157:   var LinkName: string; Scale: Integer = 100); overload;
3158: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3159:   const State: TOwnerDrawState; const Text: string; var Width, Height: Integer;
3160:   CalcType: TJvHTMLCalcType; MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3161:   var LinkName: string; Scale: Integer = 100); overload;
3162: function HTMLDrawText(Canvas: TCanvas; Rect: TRect;
3163:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): string;
3164: function HTMLDrawTextHL(Canvas: TCanvas; Rect: TRect;
3165:   const State: TOwnerDrawState; const Text: string; MouseX, MouseY: Integer;
3166:   Scale: Integer = 100): string;
3167: function HTMLPlainText(const Text: string): string;
3168: function HTMLTextExtent(Canvas: TCanvas; Rect: TRect;
3169:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): TSize;
3170: function HTMLTextWidth(Canvas: TCanvas; Rect: TRect;
3171:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): Integer;
3172: function HTMLTextHeight(Canvas: TCanvas; const Text: string; Scale: Integer = 100): Integer;
3173: function HTMLPrepareText(const Text: string): string;
3174:
3175: ***** uPSI_JvAppUtils;
3176: Function GetDefaultSection( Component : TComponent) : string
3177: Procedure GetDefaultIniData(Control:TControl; var IniFileName, Section : string; UseRegistry : Boolean)
3178: Procedure GetDefaultIniData( Control : TControl; var IniFileName, Section : string)
3179: Function GetDefaultIniName : string
3180: ///'OnGetDefaultIniName','TOnGetDefaultIniName').SetString();
3181: Function GetDefaultIniRegKey : string
3182: Function FindForm( FormClass : TFormClass) : TForm
3183: Function FindShowForm( FormClass : TFormClass; const Caption : string) : TForm
3184: Function ShowDialog( FormClass : TFormClass) : Boolean
3185: ///Function InstantiateForm( FormClass : TFormClass; var Reference) : TForm
3186: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3187: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3188: Procedure SaveMDIChildrenReg( MainForm : TForm; IniFile : TRegIniFile)
3189: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string)
3190: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string)
3191: Function GetUniqueFileNameInDir( const Path, FileNameMask : string) : string
3192: Function StrToIniStr( const Str : string) : string
3193: Function IniStrToStr( const Str : string) : string
3194: Function IniReadString( IniFile : TObjet; const Section, Ident, Default : string) : string
3195: Procedure IniWriteString( IniFile : TObjet; const Section, Ident, Value : string)
3196: Function IniReadInteger( IniFile : TObjet; const Section, Ident : string; Default : Longint) : Longint
3197: Procedure IniWriteInteger( IniFile : TObjet; const Section, Ident : string; Value : Longint)
3198: Function IniReadBool( IniFile : TObjet; const Section, Ident : string; Default : Boolean) : Boolean
3199: Procedure IniWriteBool( IniFile : TObjet; const Section, Ident : string; Value : Boolean)
3200: Procedure IniReadSections( IniFile : TObjet; Strings : TStrings)
3201: Procedure IniEraseSection( IniFile : TObjet; const Section : string)
3202: Procedure IniDeleteKey( IniFile : TObjet; const Section, Ident : string)
3203: Procedure AppBroadcast( Msg, wParam : Longint; lParam : Longint)
3204: Procedure AppBroadcast( Msg, wParam : Word; lParam : Longint)
3205: Procedure AppTaskbarIcons( AppOnly : Boolean)
3206: Procedure InternalSaveGridLayout( Grid : TCustomGrid; IniFile : TObjet; const Section : string)
3207: Procedure InternalRestoreGridLayout( Grid : TCustomGrid; IniFile : TObjet; const Section : string)
3208: Procedure InternalSaveMDIChildren( MainForm : TForm; IniFile : TObjet)
3209: Procedure InternalRestoreMDIChildren( MainForm : TForm; IniFile : TObjet)
3210: ***** uPSI_JvDBUtils;
3211: Function CreateLocate( DataSet : TDataSet) : TJvLocateObject
3212: Function IsDataSetEmpty( DataSet : TDataSet) : Boolean
3213: Procedure RefreshQuery( Query : TDataSet)
3214: Function DataSetSortedSearch(DataSet:TDataSet;const Value,FieldName:string;CaseInsensitive:Bool):Boolean
3215: Function DataSetSectionName( DataSet : TDataSet) : string
3216: Procedure InternalSaveFields( DataSet : TDataSet; IniFile : TObjet; const Section : string)
3217: Procedure InternalRestoreFields(DataSet:TDataSet;IniFile:TObjet;const Section:string;RestoreVisible:Bool)
3218: Function DataSetLocateThrough(DataSet:TDataSet; const KeyFields: string; const KeyValues: Variant;
Options: TLocateOptions) : Boolean
3219: Procedure SaveFields( DataSet : TDataSet; IniFile : TIniFile)
3220: Procedure RestoreFields( DataSet : TDataSet; IniFile : TIniFile; RestoreVisible : Boolean)
3221: Procedure AssignRecord( Source, Dest : TDataSet; ByName : Boolean)
3222: Function ConfirmDelete : Boolean
3223: Procedure ConfirmDataSetCancel( DataSet : TDataSet)
3224: Procedure CheckRequiredField( Field : TField)

```

```

3225: Procedure CheckRequiredFields( const Fields : array of TField)
3226: Function DateToSQL( Value : TDateTime) : string
3227: Function FormatSQLDateRange( Date1, Date2 : TDateTime; const FieldName : string) : string
3228: Function FormatSQLDateRangeEx( Date1, Date2 : TDateTime; const FieldName : string) : string
3229: Function FormatSQLNumericRange( const FieldName:string; LowVal, HighVal, LowEmpty,
    HighEmpty:Double; Inclusive:Bool):string
3230: Function StrMaskSQL( const Value : string) : string
3231: Function FormatSQLCondition( const FieldName, Operator, Val:string; FieldType:TFieldType; Exact:Bool):string
3232: Function FormatAnsiSQLCondition( const FieldName, Operator, Val:string; FieldType:TFieldType; Exact:Bool):string
3233: Procedure _DBError( const Msg : string)
3234: Const('TrueExpr','String').SetString( '0=0
3235: Const('sdfStandard16','String').SetString( '""mm'/'dd'/'yyyy'""
3236: Const('sdfStandard32','String').SetString( '""dd/mm/yyyy'""
3237: Const('sdfOracle','String').SetString( "TO_DATE('dd/mm/yyyy', 'DD/MM/YYYY')"
3238: Const('sdfInterbase','String').SetString( "CAST('mm'/'dd'/'yyyy' AS DATE)"
3239: Const('sdfMSSQL','String').SetString( "CONVERT(datetime, 'mm'/'dd'/'yyyy', 103)"
3240: AddTypeS('Largeint', 'Longint
3241: addTypeS('TIFException', '(ErNoError, erCannotImport, erInvalidType, ErInternalError, '+
3242: 'erInvalidHeader, erInvalidOpcode, erInvalidOpcodeParameter, erNoMainProc, erOutOfGlobalVarsRange, '+
3243: 'erOutOfProcRange, ErOutOfRange, erOutOfStackRange, ErTypeMismatch, erUnexpectedEof, '+
3244: 'erVersionError, ErDivideByZero, ErMathError, erCouldNotCallProc, erOutOfRecordRange, '+
3245: 'erOutOfMemory,erException,erNullPointerException,erNullVariantErrorerInterfaceNotSupportedError);
3246: (*-----*)
3247: procedure SIRegister_JclIniFiles(CL: TPSPascalCompiler);
3248: begin
3249: Function JIniReadBool( const FileName, Section, Line : string) : Boolean
3250: Function JIniReadInteger( const FileName, Section, Line : string) : Integer
3251: Function JIniReadString( const FileName, Section, Line : string) : string
3252: Procedure JIniWriteBool( const FileName, Section, Line : string; Value : Boolean)
3253: Procedure JIniWriteInteger( const FileName, Section, Line : string; Value : Integer)
3254: Procedure JIniWriteString( const FileName, Section, Line, Value : string)
3255: Procedure JIniReadStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3256: Procedure JIniWriteStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3257: end;
3258:
3259: (* === compile-time registration functions === *)
3260: (*-----*)
3261: procedure SIRegister_JclDateTime(CL: TPSPascalCompiler);
3262: begin
3263: 'UnixTimeStart','LongInt').SetInt( 25569);
3264: Function JEncodeDate( const Year : Integer; Month, Day : Word) : TDateTime
3265: Procedure JDecodeDate( Date : TDateTime; var Year, Month, Day : Word);
3266: Procedure DecodeDate1( Date : TDateTime; var Year : Integer; var Month, Day : Word);
3267: Procedure DecodeDate2( Date : TDateTime; var Year, Month, Day : Integer);
3268: Function CenturyOfDate( const DateTime : TDateTime) : Integer
3269: Function CenturyBaseYear( const DateTime : TDateTime) : Integer
3270: Function DayOfDate( const DateTime : TDateTime) : Integer
3271: Function MonthOfDate( const DateTime : TDateTime) : Integer
3272: Function YearOfDate( const DateTime : TDateTime) : Integer
3273: Function JDayOfTheYear( const DateTime : TDateTime; var Year : Integer) : Integer;
3274: Function DayOfTheYear1( const DateTime : TDateTime) : Integer;
3275: Function DayOfTheYearToDateTime( const Year, Day : Integer) : TDateTime
3276: Function HourOfTime( const DateTime : TDateTime) : Integer
3277: Function MinuteOfTime( const DateTime : TDateTime) : Integer
3278: Function SecondOfTime( const DateTime : TDateTime) : Integer
3279: Function GetISOYearNumberOfDays( const Year : Word) : Word
3280: Function IsISOLongYear( const Year : Word) : Boolean;
3281: Function IsISOLongYear1( const DateTime : TDateTime) : Boolean;
3282: Function ISODayOfWeek( const DateTime : TDateTime) : Word
3283: Function JISOWeekNumber( DateTime : TDateTime; var YearOfWeekNumber, WeekDay : Integer) : Integer;
3284: Function ISOWeekNumber1( DateTime : TDateTime; var YearOfWeekNumber : Integer) : Integer;
3285: Function ISOWeekNumber2( DateTime : TDateTime) : Integer;
3286: Function ISOWeekToDateTime( const Year, Week, Day : Integer) : TDateTime
3287: Function JIsLeapYear( const Year : Integer) : Boolean;
3288: Function IsLeapYear1( const DateTime : TDateTime) : Boolean;
3289: Function JDaysInMonth( const DateTime : TDateTime) : Integer
3290: Function Make4DigitYear( Year, Pivot : Integer) : Integer
3291: Function JMakeYear4Digit( Year, WindowsillYear : Integer) : Integer
3292: Function JEasterSunday( const Year : Integer) : TDateTime // TDosDateTime, 'Integer
3293: Function JFormatDateTime( Form : string; DateTime : TDateTime) : string
3294: Function FATDatesEqual( const FileTime1, FileTime2 : Int64) : Boolean;
3295: Function FATDatesEqual1( const FileTime1, FileTime2 : TFileTime) : Boolean;
3296: Function HoursToMsecs( Hours : Integer) : Integer
3297: Function MinutesToMsecs( Minutes : Integer) : Integer
3298: Function SecondsToMsecs( Seconds : Integer) : Integer
3299: Function TimeOfDateTimeToSeconds( DateTime : TDateTime) : Integer
3300: Function TimeOfDateTimeToMsecs( DateTime : TDateTime) : Integer
3301: Function DateTimeToLocalDateTime( DateTime : TDateTime) : TDateTime
3302: Function LocalDateTimeToDateTime( DateTime : TDateTime) : TDateTime
3303: Function DateTimeToDosDateTime( const DateTime : TDateTime) : TDosDateTime
3304: Function JDateTimeToFileTime( DateTime : TDateTime) : TFileTime
3305: Function JDateTimeToSystemTime( DateTime : TDateTime) : TSystemTime;
3306: Procedure DateTimeToSystemTime1( DateTime : TDateTime; var SysTime : TSystemTime);
3307: Function LocalDateTimeToFileTime( DateTime : TDateTime) : FileTime
3308: Function DosDateTimeToDateTime( const DosTime : TDosDateTime) : TDateTime
3309: Function JDosDateTimeToFileTime( DosTime : TDosDateTime) : TFileTime;
3310: Procedure DosDateTimeToFileTime1( DTH, DTL : Word; FT : TFileTime);
3311: Function DosDateTimeToSystemTime( const DosTime : TDosDateTime) : TSystemTime
3312: Function DosDateTimeToStr( DateTime : Integer) : string

```

```

3313: Function JFileTimeToDateTime( const FileTime : TFileTime) : TDateTime
3314: Function FileTimeToLocalDateTime( const FileTime : TFileTime) : TDateTime
3315: Function JFileTimeToDosDateTime( const FileTime : TFileTime) : TDosDateTime;
3316: Procedure FileTimeToDosDateTime1( const FileTime : TFileTime; var Date, Time : Word);
3317: Function JFileTimeToSystemTime( const FileTime : TFileTime) : TSystemTime;
3318: Procedure FileTimeToSystemTime1( const FileTime : TFileTime; var ST : TSystemTime);
3319: Function FileTimeToStr( const FileTime : TFileTime) : string
3320: Function SystemTimeToDosDateTime( const SystemTime : TSystemTime) : TDosDateTime
3321: Function JSystemTimeToFileTime( const SystemTime : TSystemTime) : TFileTime;
3322: Procedure SystemTimeToFileTime1( const SystemTime : TSystemTime; FTime : TFileTime);
3323: Function SystemTimeToStr( const SystemTime : TSystemTime) : string
3324: Function CreationDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3325: Function LastAccessDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3326: Function LastWriteDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3327: TJclUnixTime32, 'Longword
3328: Function JDateTimeToUnixTime( DateTime : TDateTime) : TJclUnixTime32
3329: Function JUnixTimeToDateTime( const UnixTime : TJclUnixTime32) : TDateTime
3330: Function FileTimeToUnixTime( const AValue : TFileTime) : TJclUnixTime32
3331: Function UnixTimeToFileTime( const AValue : TJclUnixTime32) : TFileTime
3332: Function JNullStamp : TTimeStamp
3333: Function JCompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
3334: Function JEqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
3335: Function JIsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
3336: Function TimeStampDOW( const Stamp : TTimeStamp) : Integer
3337: Function FirstWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3338: Function FirstWeekDay1( const Year, Month : Integer) : Integer;
3339: Function LastWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3340: Function LastWeekDay1( const Year, Month : Integer) : Integer;
3341: Function IndexedWeekDay( const Year, Month : Integer; Index : Integer) : Integer
3342: Function FirstWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3343: Function FirstWeekendDay1( const Year, Month : Integer) : Integer;
3344: Function LastWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3345: Function LastWeekendDay1( const Year, Month : Integer) : Integer;
3346: Function IndexedWeekendDay( const Year, Month : Integer; Index : Integer) : Integer
3347: Function FirstDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3348: Function LastDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3349: Function IndexedDayOfWeek( const Year, Month, DayOfWeek, Index : Integer) : Integer
3350: FindClass('TOBJECT'),'EJclDateTimeError
3351: end;
3352:
3353: procedure SIRegister_JclMiscel2(CL: TPSPascalCompiler);
3354: begin
3355:   Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
3356:   Function CreateDOSProcessRedirected( const CommandLine, InputFile, OutputFile : string) : Boolean
3357:   Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
3358:   Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
3359:   Function WinExec32AndRedirectOutput(const Cmd: string; var Output: string; RawOutput:Boolean):Cardinal
3360:   TJclKillLevel, '( klNormal, klNoSignal, klTimeout )
3361:   Function ExitWindows( ExitCode : Cardinal) : Boolean
3362:   Function LogOffOS( KillLevel : TJclKillLevel) : Boolean
3363:   Function PowerOffOS( KillLevel : TJclKillLevel) : Boolean
3364:   Function ShutDownOS( KillLevel : TJclKillLevel) : Boolean
3365:   Function RebootOS( KillLevel : TJclKillLevel) : Boolean
3366:   Function HibernateOS( Force, DisableWakeEvents : Boolean) : Boolean
3367:   Function SuspendOS( Force, DisableWakeEvents : Boolean) : Boolean
3368:   Function ShutDownDialog( const DialogMessage:string;TimeOut:DWORD;Force,Reboot:Boolean):Boolean;;
3369:   Function ShutDownDialog1(const MachineName,DialogMessage:string;TimeOut:DWORD;Force,Reboot:Bool):Bool;
3370:   Function AbortShutDown : Boolean;
3371:   Function AbortShutDown1( const MachineName : string) : Boolean;
3372:   TJclAllowedPowerOperation, '( apoHibernate, apoShutdown, apoSuspend )
3373:   TJclAllowedPowerOperations, 'set of TJclAllowedPowerOperation
3374:   Function GetAllowedPowerOperations : TJclAllowedPowerOperations
3375:   FindClass('TOBJECT'),'EJclCreateProcessError
3376:   Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
3377:   Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string; const
Environment:PChar);
3378:   // with Add(EJclCreateProcessError) do
3379:   end;
3380:
3381:
3382: procedure SIRegister_JclAnsiStrings(CL: TPSPascalCompiler);
3383: begin
3384:   //'AnsiSigns','Set').SetSet(['-', '+']);
3385:   'C1_UPPER','LongWord').SetUInt( $0001);
3386:   'C1_LOWER','LongWord').SetUInt( $0002);
3387:   'C1_DIGIT','LongWord').SetUInt( $0004);
3388:   'C1_SPACE','LongWord').SetUInt( $0008);
3389:   'C1_PUNCT','LongWord').SetUInt( $0010);
3390:   'C1_CNTRL','LongWord').SetUInt( $0020);
3391:   'C1_BLANK','LongWord').SetUInt( $0040);
3392:   'C1_XDIGIT','LongWord').SetUInt( $0080);
3393:   'C1_ALPHA','LongWord').SetUInt( $0100);
3394:   AnsiChar, 'Char
3395:   Function StrIsAlpha( const S : AnsiString) : Boolean
3396:   Function StrIsAlphaNum( const S : AnsiString) : Boolean
3397:   Function StrIsAlphaNumUnderscore( const S : AnsiString) : Boolean
3398:   Function StrContainsChars(const S:AnsiString;Chars:TSysCharSet; CheckAll : Boolean) : Boolean
3399:   Function StrConsistsOfNumberChars( const S : AnsiString) : Boolean
3400:   Function StrIsDigit( const S : AnsiString) : Boolean

```

```

3401: Function StrIsSubset( const S : AnsiString; const ValidChars : TSysCharSet) : Boolean
3402: Function StrSame( const S1, S2 : AnsiString) : Boolean
3403: //Function StrCenter( const S : AnsiString; L : Integer; C : AnsiChar) : AnsiString
3404: Function StrCharPosLower( const S : AnsiString; CharPos : Integer) : AnsiString
3405: Function StrCharPosUpper( const S : AnsiString; CharPos : Integer) : AnsiString
3406: Function StrDoubleQuote( const S : AnsiString) : AnsiString
3407: Function StrEnsureNoPrefix( const Prefix, Text : AnsiString) : AnsiString
3408: Function StrEnsureNoSuffix( const Suffix, Text : AnsiString) : AnsiString
3409: Function StrEnsurePrefix( const Prefix, Text : AnsiString) : AnsiString
3410: Function StrEnsureSuffix( const Suffix, Text : AnsiString) : AnsiString
3411: Function StrEscapedToString( const S : AnsiString) : AnsiString
3412: Function JStrLower( const S : AnsiString) : AnsiString
3413: Procedure StrLowerInPlace( var S : AnsiString)
3414: ///Procedure StrLowerBuff( S : PAnsiChar)
3415: Procedure JStrMove( var Dest:AnsiString; const Source:AnsiString;const ToIndex,FromIndex,Count:Integer;
3416: Function StrPadLeft( const S : AnsiString; Len : Integer; C : AnsiChar) : AnsiString
3417: Function StrPadRight( const S : AnsiString; Len : Integer; C : AnsiChar) : AnsiString
3418: Function StrProper( const S : AnsiString) : AnsiString
3419: //Procedure StrProperBuff( S : PAnsiChar)
3420: Function StrQuote( const S : AnsiString; C : AnsiChar) : AnsiString
3421: Function StrRemoveChars( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3422: Function StrKeepChars( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3423: Procedure JStrReplace(var S:AnsiString; const Search, Replace : AnsiString; Flags : TReplaceFlags)
3424: Function StrReplaceChar( const S : AnsiString; const Source, Replace : AnsiChar) : AnsiString
3425: Function StrReplaceChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString
3426: Function StrReplaceButChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString;
3427: Function StrRepeat( const S : AnsiString; Count : Integer) : AnsiString
3428: Function StrRepeatLength( const S : AnsiString; const L : Integer) : AnsiString
3429: Function StrReverse( const S : AnsiString) : AnsiString
3430: Procedure StrReverseInPlace( var S : AnsiString)
3431: Function StrSingleQuote( const S : AnsiString) : AnsiString
3432: Function StrSmartCase( const S : AnsiString; Delimiters : TSysCharSet) : AnsiString
3433: Function StrStringToEscaped( const S : AnsiString) : AnsiString
3434: Function StrStripNonNumberChars( const S : AnsiString) : AnsiString
3435: Function StrToHex( const Source : AnsiString) : AnsiString
3436: Function StrTrimCharLeft( const S : AnsiString; C : AnsiChar) : AnsiString
3437: Function StrTrimCharsLeft( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3438: Function StrTrimCharRight( const S : AnsiString; C : AnsiChar) : AnsiString
3439: Function StrTrimCharsRight( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3440: Function StrTrimQuotes( const S : AnsiString) : AnsiString
3441: Function JStrUpper( const S : AnsiString) : AnsiString
3442: Procedure StrUpperInPlace( var S : AnsiString)
3443: ///Procedure StrUpperBuff( S : PAnsiChar)
3444: Function StrOemToAnsi( const S : AnsiString) : AnsiString
3445: Function StrAnsiToOem( const S : AnsiString) : AnsiString
3446: Procedure StrAddRef( var S : AnsiString)
3447: Function StrAllocSize( const S : AnsiString) : Longint
3448: Procedure StrDecRef( var S : AnsiString)
3449: //Function StrLen( S : PAnsiChar) : Integer
3450: Function StrLength( const S : AnsiString) : Longint
3451: Function StrRefCount( const S : AnsiString) : Longint
3452: Procedure StrResetLength( var S : AnsiString)
3453: Function StrCharCount( const S : AnsiString; C : AnsiChar) : Integer
3454: Function StrCharsCount( const S : AnsiString; Chars : TSysCharSet) : Integer
3455: Function StrStrCount( const S, SubS : AnsiString) : Integer
3456: Function StrCompare( const S1, S2 : AnsiString) : Integer
3457: Function StrCompareRange( const S1, S2 : AnsiString; const Index, Count : Integer) : Integer
3458: //Function StrFillChar( const C : AnsiChar; Count : Integer) : AnsiString;
3459: Function StrFillChar( const C : Char; Count : Integer) : AnsiString;
3460: Function StrFillChar(const C: Char; Count: Integer): string'';
3461: Function IntFillChar(const I: Integer; Count: Integer): string'';
3462: Function ByteFillChar(const B: Byte; Count: Integer): string'';
3463: Function ArrFillChar(const AC: Char; Count: Integer): TCharArray;'';
3464: Function ArrByteFillChar(const AB: Char; Count: Integer): TByteArray;
3465: Function StrFind( const Substr, S : AnsiString; const Index : Integer) : Integer
3466: //Function StrHasPrefix( const S : AnsiString; const Prefixes : array of AnsiString) : Boolean
3467: Function StrIndex( const S : AnsiString; const List : array of AnsiString) : Integer
3468: Function StrILastPos( const SubStr, S : AnsiString) : Integer
3469: Function StrIPos( const SubStr, S : AnsiString) : Integer
3470: Function StrIsOneOf( const S : AnsiString; const List : array of AnsiString) : Boolean
3471: Function StrLastPos( const SubStr, S : AnsiString) : Integer
3472: Function StrMatch( const Substr, S : AnsiString; const Index : Integer) : Integer
3473: Function StrMatches( const Substr, S : AnsiString; const Index : Integer) : Boolean
3474: Function StrNIPos( const S, SubStr : AnsiString; N : Integer) : Integer
3475: Function StrNPos( const S, SubStr : AnsiString; N : Integer) : Integer
3476: Function StrPrefixIndex( const S : AnsiString; const Prefixes : array of AnsiString) : Integer
3477: Function StrSearch( const Substr, S : AnsiString; const Index : Integer) : Integer
3478: //Function StrAfter( const SubStr, S : AnsiString) : AnsiString
3479: //Function StrBefore( const SubStr, S : AnsiString) : AnsiString
3480: Function StrBetween( const S : AnsiString; const Start, Stop : AnsiChar) : AnsiString
3481: Function StrChopRight( const S : AnsiString; N : Integer) : AnsiString
3482: Function StrLeft( const S : AnsiString; Count : Integer) : AnsiString
3483: Function StrMid( const S : AnsiString; Start, Count : Integer) : AnsiString
3484: Function StrRestOf( const S : AnsiString; N : Integer) : AnsiString
3485: Function StrRight( const S : AnsiString; Count : Integer) : AnsiString
3486: Function CharEqualNoCase( const C1, C2 : AnsiChar) : Boolean
3487: Function CharIsAlpha( const C : AnsiChar) : Boolean
3488: Function CharIsAlphaNum( const C : AnsiChar) : Boolean
3489: Function CharIsBlank( const C : AnsiChar) : Boolean

```



```

3490: Function CharIsControl( const C : AnsiChar) : Boolean
3491: Function CharIsDelete( const C : AnsiChar) : Boolean
3492: Function CharIsDigit( const C : AnsiChar) : Boolean
3493: Function CharIsLower( const C : AnsiChar) : Boolean
3494: Function CharIsNumberChar( const C : AnsiChar) : Boolean
3495: Function CharIsPrintable( const C : AnsiChar) : Boolean
3496: Function CharIsPunctuation( const C : AnsiChar) : Boolean
3497: Function CharIsReturn( const C : AnsiChar) : Boolean
3498: Function CharIsSpace( const C : AnsiChar) : Boolean
3499: Function CharIsUpper( const C : AnsiChar) : Boolean
3500: Function CharIsWhiteSpace( const C : AnsiChar) : Boolean
3501: Function CharType( const C : AnsiChar) : Word
3502: Function CharHex( const C : AnsiChar) : Byte
3503: Function CharLower( const C : AnsiChar) : AnsiChar
3504: Function CharUpper( const C : AnsiChar) : AnsiChar
3505: Function CharToggleCase( const C : AnsiChar) : AnsiChar
3506: Function CharPos( const S : AnsiString; const C : AnsiChar; const Index : Integer) : Integer
3507: Function CharLastPos( const S : AnsiString; const C : AnsiChar; const Index : Integer) : Integer
3508: Function CharIPos( const S : AnsiString; C : AnsiChar; const Index : Integer) : Integer
3509: Function CharReplace( var S : AnsiString; const Search, Replace : AnsiChar) : Integer
3510: Procedure StrToTStrings( S, Sep : AnsiString; const List : TStrings; const AllowEmptyString : Boolean)
3511: Procedure StrToTStrings( S, Sep : AnsiString; const List : TStrings; const AllowEmptyString : Boolean)
3512: Function StringsToStr(const List:TStrings;const Sep:AnsiString;const AllowEmptyString:Bool):AnsiString;
3513: Procedure TrimStrings( const List : TStrings; DeleteIfEmpty : Boolean)
3514: Procedure TrimStringsRight( const List : TStrings; DeleteIfEmpty : Boolean)
3515: Procedure TrimStringsLeft( const List : TStrings; DeleteIfEmpty : Boolean)
3516: Function AddStringToTStrings(const S:AnsiString;Strings:TStrings; const Unique:Boolean):Boolean
3517: Function BooleanToStr( B : Boolean) : AnsiString
3518: Function FileToString( const FileName : AnsiString) : AnsiString
3519: Procedure StringToFile( const FileName, Contents : AnsiString; Append : Boolean)
3520: Function StrToken( var S : AnsiString; Separator : AnsiChar) : AnsiString
3521: Procedure StrTokens( const S : AnsiString; const List : TStrings)
3522: Procedure StrTokenToTStrings( S : AnsiString; Separator : AnsiChar; const List : TStrings)
3523: //Function StrWord( var S : PAnsiChar; out Word : AnsiString) : Boolean
3524: Function StrToFloatSafe( const S : AnsiString) : Float
3525: Function StrToIntSafe( const S : AnsiString) : Integer
3526: Procedure StrNormIndex( const StrLen : Integer; var Index : Integer; var Count : Integer);
3527: Function ArrayOf( List : TStrings) : TDynStringArray;
3528: EJclStringError', 'EJclError
3529: function IsClass(Address: TObject): Boolean;
3530: function IsObject(Address: TObject): Boolean;
3531: // Console Utilities
3532: //function ReadKey: Char;
3533: function IntToStrZeroPad(Value, Count: Integer): AnsiString;
3534: function JclGUIDToString(const GUID: TGUID): string;
3535: function JclStringToGUID(const S: string): TGUID;
3536:
3537: end;
3538:
3539:
3540: ***** uPSI_JvDBUtil;
3541: Procedure ExecuteSQLScript(Base:TDataBase; const Script: string; const
Commit:TCommit;OnProgress:TOnProgress; const UserData : Integer)
3542: Function GetQueryResult( const DatabaseName, SQL : string) : Variant
3543: Function GetStoredProcResult( const ADatabaseName, AStoredProcName : string; AParams : array of Variant;
const AResultName : string) : Variant
3544: //Function StrFieldDesc( Field : FLDDesc) : string
3545: Function Var2Type( V : Variant; const VarType : Integer) : Variant
3546: Procedure CopyRecord( DataSet : TDataSet)
3547: //Procedure AddReference( Tbl : TTable; RefName : string; RefField : Word; MasterTable : string;
MasterField : Word; ModOp, DelOp : RINTQual)
3548: Procedure AddMasterPassword( Table : TTable; pswd : string)
3549: Procedure PackTable( Table : TTable)
3550: Procedure PackEncryptedTable( Table : TTable; pswd : string)
3551: Function EncodeQuotes( const S : string) : string
3552: Function Cmp( const S1, S2 : string) : Boolean
3553: Function SubStr( const S : string; const Index : Integer; const Separator : string) : string
3554: Function SubStrEnd( const S : string; const Index : Integer; const Separator : string) : string
3555: Function ReplaceString( S : string; const OldPattern, NewPattern : string) : string
3556: Procedure GetXYByPos( const S : string; const Pos : Integer; var X, Y : Integer)
3557: *****uPSI_JvJvBDEUtils;*****
3558: //JvBDEUtils
3559: Function CreateDbLocate : TJvLocateObject
3560: //Function CheckOpen( Status : DBIResult) : Boolean
3561: Procedure FetchAllRecords( DataSet : TBDEDataSet)
3562: Function TransActive( Database : TDatabase) : Boolean
3563: Function AsyncQrySupported( Database : TDatabase) : Boolean
3564: Function GetQuoteChar( Database : TDatabase) : string
3565: Procedure ExecuteQuery( const DbName, QueryText : string)
3566: Procedure ExecuteQueryEx( const SessName, DbName, QueryText : string)
3567: Function FieldLogicMap( FldType : TFieldType) : Integer
3568: Function FieldSubtypeMap( FldType : TFieldType) : Integer Value : string; Buffer : Pointer)
3569: Function GetAliasPath( const AliasName : string) : string
3570: Function IsDirectory( const DatabaseName : string) : Boolean
3571: Function GetBdeDirectory : string
3572: Function LoginToDatabase( Database : TDatabase; OnLogin : TDatabaseLoginEvent) : Boolean
3573: Function DataSetFindValue( ADataSet : TBDEDataSet; const Value, FieldName : string) : Boolean
3574: Function DataSetFindLike( ADataSet : TBDEDataSet; const Value, FieldName : string) : Boolean
3575: Function DataSetRecNo( DataSet : TDataSet) : Longint

```

```

3576: Function DataSetRecordCount( DataSet : TDataSet ) : Longint
3577: Function DataSetPositionStr( DataSet : TDataSet ) : string
3578: Procedure DataSetShowDeleted( DataSet : TBDEDataSet; Show : Boolean)
3579: Function CurrentRecordDeleted( DataSet : TBDEDataSet ) : Boolean
3580: Function IsFilterApplicable( DataSet : TDataSet ) : Boolean
3581: Function IsBookmarkStable( DataSet : TBDEDataSet ) : Boolean
3582: Procedure SetIndex( Table : TTable; const IndexFieldNames : string)
3583: Procedure RestoreIndex( Table : TTable)
3584: Procedure DeleteRange( Table : TTable; IndexFields : array of const; FieldValues : array of const)
3585: Procedure PackTable( Table : TTable)
3586: Procedure ReindexTable( Table : TTable)
3587: Procedure BdeFlushBuffers
3588: Function GetNativeHandle( Database : TDatabase; Buffer : Pointer; BufSize : Integer ) : Pointer
3589: Procedure ToggleDebugLayer( Active : Boolean; const DebugFile : string)
3590: Procedure DbNotSupported
3591: Procedure ExportDataSet( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
  AsciiCharSet : string; AsciiDelimited : Boolean; MaxRecordCount : Longint)
3592: Procedure ExportDataSetEx( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
  AsciiCharSet:string;AsciiDelimited:Boolean;AsciiDelimiter:Char;MaxRecordCount:Longint);
3593: Procedure
  ImportDataSet(Source:TBDEDataSet;DestTable:TTable;MaxRecordCount:Longint;Mappings:TStrings;Mode:TBatchMode);
3594: Procedure InitRSRUN(Database: TDatabase;const ConName:string; ConType:Integer;const ConServer:string);
3595: *****uPSI_JvDateUtil;
3596: function CurrentYear: Word;
3597: function IsLeapYear(AYear: Integer): Boolean;
3598: function DaysPerMonth(AYear, AMonth: Integer): Integer;
3599: function FirstDayOfPrevMonth: TDateTime;
3600: function LastDayOfPrevMonth: TDateTime;
3601: function FirstDayOfNextMonth: TDateTime;
3602: function ExtractDay(ADate: TDateTime): Word;
3603: function ExtractMonth(ADate: TDateTime): Word;
3604: function ExtractYear(ADate: TDateTime): Word;
3605: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
3606: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime;
3607: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
3608: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
3609: function ValidDate(ADate: TDateTime): Boolean;
3610: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
3611: function MonthsBetween(Date1, Date2: TDateTime): Double;
3612: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
3613: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
3614: function DaysBetween(Date1, Date2: TDateTime): Longint;
3615: { The same as previous but if Date2 < Date1 result = 0 }
3616: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
3617: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
3618: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
3619: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
3620: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
3621: function CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
3622: { String to date conversions }
3623: function GetDateOrder(const DateFormat: string): TDateOrder;
3624: function MonthFromName(const S: string; MaxLen: Byte): Byte;
3625: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
3626: function StrToDateFmt(const DateFormat, S: string): TDateTime;
3627: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
3628: function DefDateFormat(FourDigitYear: Boolean): string;
3629: function DefDateMask(BlanksChar: Char; FourDigitYear: Boolean): string;
3630: -----
3631: ***** JvUtils;*****
3632: { GetWordOnPos returns Word from string, S, on the cursor position, P }
3633: function GetWordOnPos(const S: string; const P: Integer): string;
3634: { GetWordOnPosEx work like GetWordOnPos function, also returns Word position in iBeg, iEnd variables }
3635: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3636: { SubStr returns substring from string, S, separated with Separator string }
3637: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3638: { SubStrEnd same to previous function but Index numerated from the end of string }
3639: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3640: { SubWord returns next word from string, P, and offsets Pointer to the end of Word, P2 }
3641: function SubWord(P: PChar; var P2: PChar): string;
3642: { NumberByWord returns the text representation of
  the number, N, in normal russian language. Was typed from Monitor magazine }
3644: function NumberByWord(const N: Longint): string;
3645: // function CurrencyByWord(Value : Currency) : string; GetLineByPos returns Line number, there
3646: //the symbol Pos is pointed. Lines separated with #13 symbol }
3647: function GetLineByPos(const S: string; const Pos: Integer): Integer;
3648: { GetXYByPos is same to previous function, but returns X position in line too }
3649: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3650: { ReplaceString searches for all substrings, OldPattern,in a string, S, replaces them with NewPattern }
3651: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3652: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3653: function ConcatSep(const S, S2, Separator: string): string;
3654: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
3655: function ConcatLeftSep(const S, S2, Separator: string): string;
3656: { MinimizeString truncs long string, S, and appends '...' symbols, if Length of S is more than MaxLen }
3657: function MinimizeString(const S: string; const MaxLen: Integer): string;
3658: { Next 4 function for russian chars transliterating.
  This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes works sucks }
3660: procedure Dos2Win(var S: string);
3661: procedure Win2Dos(var S: string);

```

```

3662: function Dos2WinRes(const S: string): string;
3663: function Win2DosRes(const S: string): string;
3664: function Win2Koi(const S: string): string;
3665: { Spaces returns string consists on N space chars }
3666: function Spaces(const N: Integer): string;
3667: { AddSpaces add spaces to string, S, if it Length is smaller than N }
3668: function AddSpaces(const S: string; const N: Integer): string;
3669: { function LastDate for russian users only } { returns date relative to current date: ' ' }
3670: function LastDate(const Dat: TDateTime): string;
3671: { CurrencyToStr format currency, Cur, using ffCurrency float format}
3672: function CurrencyToStr(const Cur: currency): string;
3673: { Cmp compares two strings and returns True if they are equal. Case-insensitive.}
3674: function Cmp(const S1, S2: string): Boolean;
3675: { StringCat add S2 string to S1 and returns this string }
3676: function StringCat(var S1: string; S2: string): string;
3677: { HasChar returns True, if Char, Ch, contains in string, S }
3678: function HasChar(const Ch: Char; const S: string): Boolean;
3679: function HasAnyChar(const Chars: string; const S: string): Boolean;
3680: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3681: function CountOfChar(const Ch: Char; const S: string): Integer;
3682: function DefStr(const S: string; Default: string): string;
3683: {**** files routines}
3684: { GetWinDir returns Windows folder name }
3685: function GetWinDir: TFileName;
3686: function GetSysDir: string;
3687: { GetTempDir returns Windows temporary folder name }
3688: function GetTempDir: string;
3689: { GenTempFileName returns temporary file name on drive, there FileName is placed }
3690: function GenTempFileName(FileName: string): string;
3691: { GenTempFileNameExt same to previous function, but returning filename has given extension, FileExt }
3692: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
3693: { ClearDir clears folder Dir }
3694: function ClearDir(const Dir: string): Boolean;
3695: { DeleteDir clears and than delete folder Dir }
3696: function DeleteDir(const Dir: string): Boolean;
3697: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3698: function FileEquMask(FileName, Mask: TFileName): Boolean;
3699: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3700:   Masks must be separated with comma (','')}
3701: function FileEquMasks(FileName, Masks: TFileName): Boolean;
3702: procedure DeleteFiles(const Folder: TFileName; const Masks: string);
3703: { LZFileExpand expand file, FileSource into FileDest.File must be compressed,using MS Compress program }
3704: function LZFileExpand(const FileSource, FileDest: string): Boolean;
3705: { FileGetInfo fills SearchRec record for specified file attributes}
3706: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
3707: { HasSubFolder returns True, if folder APath contains other folders }
3708: function HasSubFolder(APath: TFileName): Boolean;
3709: { IsEmptyFolder returns True, if there are no files or folders in given folder, APath}
3710: function IsEmptyFolder(APath: TFileName): Boolean;
3711: { AddSlash add slash Char to Dir parameter, if needed }
3712: procedure AddSlash(var Dir: TFileName);
3713: { AddSlash returns string with added slash Char to Dir parameter, if needed }
3714: function AddSlash2(const Dir: TFileName): string;
3715: { AddPath returns FileName with Path, if FileName not contain any path }
3716: function AddPath(const FileName, Path: TFileName): TFileName;
3717: function AddPaths(const PathList, Path: string): string;
3718: function ParentPath(const Path: TFileName): TFileName;
3719: function FindInPath(const FileName, PathList: string): TFileName;
3720: function FindInPaths(const fileName,paths: String): String;
3721: {$IFDEF BCBI}
3722: { BrowseForFolder displays Browse For Folder dialog }
3723: function BrowseForFolder(const Handle: HWND; const Title: string; var Folder: string): Boolean;
3724: {$ENDIF BCBI}
3725: function BrowseForFolder(const ATitle: string; AllowCreate: Boolean; var ADirectory: string;
  AHelpContext: THelpContext): Boolean
3726: function BrowseForComputer(const ATitle: string; AllowCreate: Boolean; var ADirectory: string;
  AHelpContext: THelpContext): Boolean
3727: function BrowseDirectory(var AFolderName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3728: function BrowseComputer(var AComputerName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3729:
3730: { DeleteReadOnlyFile clears R/O file attribute and delete file }
3731: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
3732: { HasParam returns True, if program running with specified parameter, Param }
3733: function HasParam(const Param: string): Boolean;
3734: function HasSwitch(const Param: string): Boolean;
3735: function Switch(const Param: string): string;
3736: { ExePath returns ExtractFilePath(ParamStr(0)) }
3737: function ExePath: TFileName;
3738: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
3739: function FileTimeToDateTime(const FT: TFileTime): TDateTime;
3740: function MakeValidFileName(const FileName: TFileName; const ReplaceBadChar: Char): TFileName;
3741: {**** Graphic routines }
3742: { TTFontSelected returns True, if True Type font is selected in specified device context }
3743: function TTFontSelected(const DC: HDC): Boolean;
3744: { TrueInflateRect inflates rect in other method, than InflateRect API function }
3745: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
3746: {**** Windows routines }
3747: { SetWindowTop put window to top without recreating window }
3748: procedure SetWindowTop(const Handle: HWND; const Top: Boolean);

```

```

3749: {**** other routines }
3750: { KeyPressed returns True, if Key VK is now pressed }
3751: function KeyPressed(VK: Integer): Boolean;
3752: procedure SwapInt(var Int1, Int2: Integer);
3753: function IntPower(Base, Exponent: Integer): Integer;
3754: function ChangeTopException(E: TObject): TObject;
3755: function StrToBool(const S: string): Boolean;
3756: {$IFDEF COMPILER3_UP}
3757: { AnsiStrLIComp compares S1 to S2, without case-sensitivity, up to a maximum
3758:   Length of MaxLen bytes. The compare operation is controlled by the
3759:   current Windows locale. The return value is the same as for CompareStr. }
3760: function AnsiStrLIComp(S1, S2: PChar; MaxLen: Cardinal): Integer;
3761: function AnsiStrIComp(S1, S2: PChar): Integer;
3762: {$ENDIF}
3763: function Var2Type(V: Variant; const VarType: Integer): Variant;
3764: function VarToInt(V: Variant): Integer;
3765: function VarToFloat(V: Variant): Double;
3766: { following functions are not documented because they are don't work properly , so don't use them }
3767: function ReplaceSokrl(S: string; const Word, Frase: string): string;
3768: { ReplaceSokrl is full equal to ReplaceString function - only for compatibility - don't use }
3769: { GetSubStr is full equal to SubStr function - only for compatibility - don't use }
3770: function GetSubStr(const S: string; const Index: Integer; const Separator: Char): string;
3771: function GetParameter: string;
3772: function GetLongFileName(FileName: string): string;
3773: {* from FileCtrl}
3774: function DirectoryExists(const Name: string): Boolean;
3775: procedure ForceDirectories(Dir: string);
3776: {# from FileCtrl}
3777: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
3778: function GetComputerID: string;
3779: function GetComputerName: string;
3780: {**** string routines }
3781: { ReplaceAllSokr searches for all substrings, Words, in a string, S, and replaces them with Frases with the
   same Index. Also see RAUtilsW.ReplaceSokrl function }
3782: function ReplaceAllSokr(S: string; Words, Frases: TStrings): string;
3783: { ReplaceSokr searches the Word in a string, S, on PosBeg position,
3784:   in the list, Words, and if founds, replaces this Word with string from another list, Frases, with the
   same Index, and then update NewSelStart variable }
3785: function ReplaceSokr(S: string; PosBeg, Len: Integer; Words, Frases: TStrings; var NewSelStart: Integer): string;
3786: { CountOfLines calculates the lines count in a string, each line separated from another with CrLf sequence }
3787: function CountOfLines(const S: string): Integer;
3788: { DeleteEmptyLines deletes all empty lines from strings, Ss. Lines contained only spaces also deletes. }
3789: procedure DeleteEmptyLines(Ss: TStrings);
3790: { SQLAddWhere adds or modifies existing where-statement, where, to the strings, SQL.
3791:   Note: If strings SQL already contains where-statement, it must be started on begining of any line }
3792: procedure SQLAddWhere(SQL: TStrings; const Where: string);
3793: {**** files routines - }
3794: { ResSaveToFile save resource named as Name with Typ type into file FileName.
3795:   Resource can be compressed using MS Compress program}
3796: function ResSaveToFile(const Typ, Name: string; const Compressed: Boolean; const FileName: string): Boolean;
3797: function ResSaveToFileEx(Inst: HINST; Typ, Name: PChar; const Compressed: Boolean; const FileName: string): Boolean;
3798: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
3799: { Execute executes other program and waiting for it terminating, then return its Exit Code }
3800: function ExecuteJ(const CommandLine, WorkingDirectory: string): Integer;
3801: { IniReadSection read section, Section, from ini-file,
3802:   IniFileName, into strings, Ss. This function reads ALL strings from specified section.
3803:   Note: TIniFile.ReadSection function reads only strings with '=' symbol.}
3804: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
3805: { LoadTextFile load text file, FileName, into string }
3806: function LoadTextFile(const FileName: TFileName): string;
3807: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3808: { ReadFolder reads files list from disk folder, Folder, that are equal Mask, into strings, FileList}
3809: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
3810: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
3811: {$IFDEF COMPILER3_UP}
3812: { TargetFileName - if FileName is ShortCut returns filename ShortCut linked to }
3813: function TargetFileName(const FileName: TFileName): TFileName;
3814: { return filename ShortCut linked to }
3815: function ResolveLink(const hWnd: HWND; const LinkFile: TFileName; var FileName: TFileName): HRESULT;
3816: {$ENDIF COMPILER3_UP}
3817: {**** Graphic routines - }
3818: { LoadIcoToImage loads two icons from resource named NameRes, into two image lists ALarge and ASmall}
3819: procedure LoadIcoToImage(ALarge, ASmall: TImageList; const NameRes: string);
3820: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
3821: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
3822: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
3823: function RATextOutEx(Canvas: TCanvas; const R, RClip: TRect; const S: string; const CalcHeight: Boolean): Integer;
3824: { RATextCalcHeight calculate needed height to correct output, using RATextOut or RATextOutEx functions }
3825: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
3826: { Cinema draws some visual effect }
3827: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
3828: { Roughed fills rect with special 3D pattern }
3829: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
3830: { BitmapFromBitmap creates new small bitmap from part of source bitmap, SrcBitmap, with specified width
   and height, AWidth, AHeight and placed on a specified Index, Index in the source bitmap }
3831: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
3832: { TextWidth calculate text with for writing using standard desktop font }
3833: function TextWidth(AStr: string): Integer;
3834: { DefineCursor load cursor from resource, and return available cursor number, assigned to it }

```



```

3835: function DefineCursor(Identifier: PChar): TCursor;
3836: {**** other routines - }
3837: { FindFormByClass returns first form specified class, FormClass, owned by Application global variable }
3838: function FindFormByClass(FormClass: TFormClass): TForm;
3839: function FindFormByClassName(FormClassName: string): TForm;
3840: { FindByTag returns the control with specified class, ComponentClass, from WinControl.Controls property,
3841:   having Tag property value, equal to Tag parameter }
3842: function FindByTag(WinControl: TWinControl; ComponentClass: TComponentClass; const Tag: Integer): TComponent;
3843: { ControlAtPos2 equal to TWinControl.ControlAtPos function, but works better }
3844: function ControlAtPos2(Parent: TWinControl; X, Y: Integer): TControl;
3845: { RBTAG searches WinControl.Controls for checked RadioButton and returns its Tag property value }
3846: function RBTAG(Parent: TWinControl): Integer;
3847: { AppMinimized returns True, if Application is minimized }
3848: function AppMinimized: Boolean;
3849: { MessageBox is Application.MessageBox with string (not PChar) parameters.
3850:   if Caption parameter = '', it replaced with Application.Title }
3851: function MessageBoxJ(const Msg: string; Caption: string; const Flags: Integer): Integer;
3852: function MsgDlg2(const Msg, ACaption: string; DlgType: TMsgDlgType;
3853:   Buttons: TMsgDlgButtons; HelpContext: Integer; Control: TWinControl): Integer;
3854: function MsgDlgDef(const Msg, ACaption: string; DlgType: TMsgDlgType;
3855:   Buttons: TMsgDlgButtons; DefButton: TMsgDlgBtn; HelpContext: Integer; Control: TWinControl): Integer;
3856: { Delay stop program execution to MSec msec }
3857: procedure Delay(MSec: Longword);
3858: procedure CenterHor(Parent: TControl; MinLeft: Integer; Controls: array of TControl);
3859: procedure EnableControls(Control: TWinControl; const Enable: Boolean);
3860: procedure EnableMenuItems(MenuItem: TMenuItem; const Tag: Integer; const Enable: Boolean);
3861: procedure ExpandWidth(Parent: TControl; MinWidth: Integer; Controls: array of TControl);
3862: function PanelBorder(Panel: TCustomPanel): Integer;
3863: function Pixels(Control: TControl; APixels: Integer): Integer;
3864: procedure SetChildPropOrd(Owner: TComponent; PropName: string; Value: Longint);
3865: procedure Error(const Msg: string);
3866: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
3867:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
3868: {ex. Text parameter: 'Item 1 <b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green <c:blue>blue</i>' }
3869: function ItemHtDraw(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
3870:   const HideSelColor: Boolean): string;
3871: function ItemHtWidth(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
3872:   const HideSelColor: Boolean): Integer;
3873: function ItemHtPlain(const Text: string): string;
3874: { ClearList - clears list of TObject }
3875: procedure ClearList(List: TList);
3876: procedure MemStreamToClipboard(MemStream: TMemoryStream; const Format: Word);
3877: procedure ClipboardToMemStream(MemStream: TMemoryStream; const Format: Word);
3878: { RTTI support }
3879: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
3880: function GetPropStr(Obj: TObject; const PropName: string): string;
3881: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
3882: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
3883: procedure PrepareIniSection(SS: TStrings);
3884: { following functions are not documented because they are don't work properly, so don't use them }
3885: {$IFDEF COMPILER2}
3886: function CompareMem(P1, P2: Pointer; Length: Integer): Boolean; assembler;
3887: {$ENDIF}
3888:
3889: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
3890: begin
3891:   Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)
3892:   Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)
3893:   Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y: Int;State:TDragState;var
Accept:Bool;Sorted:Bool;
3894:   Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)
3895:   Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)
3896:   Procedure BoxSetItem( List : TWinControl; Index : Integer)
3897:   Function BoxGetFirstSelection( List : TWinControl ) : Integer
3898:   Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer ) : Boolean
3899: end;
3900:
3901: procedure SIRegister_JvCsvParse(CL: TPSPascalCompiler);
3902: begin
3903:   Const('MaxInitStrNum','LongInt').SetInt( 9);
3904:   Function JvAnsiStrSplit( const InStr: AnsiString; const SplitChar, QuoteChar:AnsiChar; var OutStrings
: array of AnsiString; MaxSplit : Integer ) : Integer
3905:   Function JvStrSplit( const InStr: string; const SplitChar, QuoteChar : Char; var OutStrings : array of
string; MaxSplit : Integer ) : Integer
3906:   Function JvAnsiStrSplitStrings(const InStr:AnsiString;const SplitChar,
QuoteChar:AnsiChar;OutStrs:TStrings):Integer;
3907:   Function JvAnsiStrStrip( S : AnsiString) : AnsiString
3908:   Function JvStrStrip( S : string ) : string
3909:   Function GetString( var Source : AnsiString; const Separator : AnsiString) : AnsiString
3910:   Function PadString( const S : AnsiString; Len : Integer; PadChar : AnsiChar ) : AnsiString
3911:   Function BuildPathName( const PathName, FileName : AnsiString) : AnsiString
3912:   Function StrEatWhiteSpace( const S : string) : string
3913:   Function HexToAscii( const S : AnsiString) : AnsiString
3914:   Function AsciiToHex( const S : AnsiString) : AnsiString
3915:   Function StripQuotes( const S1 : AnsiString) : AnsiString
3916:   Function ValidNumericLiteral( S1 : PAnsiChar) : Boolean
3917:   Function ValidIntLiteral( S1 : PAnsiChar) : Boolean
3918:   Function ValidHexLiteral( S1 : PAnsiChar) : Boolean
3919:   Function HexPCharToInt( S1 : PAnsiChar) : Integer

```

```

3920: Function ValidStringLiteral( S1 : PAnsiChar ) : Boolean
3921: Function StripPCharQuotes( S1 : PAnsiChar ) : AnsiString
3922: Function JvValidIdentifierAnsi( S1 : PAnsiChar ) : Boolean
3923: Function JvValidIdentifier( S1 : String ) : Boolean
3924: Function JvEndChar( X : AnsiChar ) : Boolean
3925: Procedure JvGetToken( S1, S2 : PAnsiChar )
3926: Function IsExpressionKeyword( S1 : PAnsiChar ) : Boolean
3927: Function IsKeyword( S1 : PAnsiChar ) : Boolean
3928: Function JvValidVarReference( S1 : PAnsiChar ) : Boolean
3929: Function GetParenthesis( S1, S2 : PAnsiChar ) : Boolean
3930: Procedure JvGetVarReference( S1, S2, SIdx : PAnsiChar )
3931: Procedure JvEatWhitespaceChars( S1 : PAnsiChar );
3932: Procedure JvEatWhitespaceChars1( S1 : PWideChar );
3933: Function GetTokenCount : Integer
3934: Procedure ResetTokenCount
3935: end;
3936:
3937: procedure SIRegister_JvDBQueryParamsForm(CL: TPSPascalCompiler);
3938: begin
3939:   SIRegister_TJvQueryParamsDialog(CL);
3940:   Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext ) : Boolean
3941:   end;
3942:
3943: ***** JvStrUtil / JvStrUtils;*****
3944: function FindNotBlankCharPos(const S: string): Integer;
3945: function AnsiChangeCase(const S: string): string;
3946: function GetWordOnPos(const S: string; const P: Integer): string;
3947: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3948: function Cmp(const S1, S2: string): Boolean;
3949: { Spaces returns string consists on N space chars }
3950: function Spaces(const N: Integer): string;
3951: { HasChar returns True, if char, Ch, contains in string, S }
3952: function HasChar(const Ch: Char; const S: string): Boolean;
3953: function HasAnyChar(const Chars: string; const S: string): Boolean;
3954: { SubStr returns substring from string, S, separated with Separator string }
3955: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3956: { SubStrEnd same to previous function but Index numerated from the end of string }
3957: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3958: { ReplaceString searches for all substrings, OldPattern, in a string, S, replaces them with NewPattern }
3959: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3960: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3961: { GetXYByPos is same to previous function, but returns X position in line too }
3962: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3963: { AddSlash returns string with added slash char to Dir parameter, if needed }
3964: function AddSlash2(const Dir: TFileName): string;
3965: { AddPath returns FileName with Path, if FileName not contain any path }
3966: function AddPath(const FileName, Path: TFileName): TFileName;
3967: { ExePath returns ExtractFilePath(ParamStr(0)) }
3968: function ExePath: TFileName;
3969: function LoadTextFile(const FileName: TFileName): string;
3970: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3971: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3972: function ConcatSep(const S, S2, Separator: string): string;
3973: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3974: function FileEquMask(FileName, Mask: TFileName): Boolean;
3975: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3976:   Masks must be separated with comma (',' ) }
3977: function FileEquMasks(FileName, Masks: TFileName): Boolean;
3978: function StringEndsWith(const Str, SubStr: string): Boolean;
3979: function ExtractFilePath2(const FileName: string): string;
3980: function StrToOem(const AnsiStr: string): string;
3981: { StrToOem translates a string from the Windows character set into the OEM character set. }
3982: function OemToAnsiStr(const OemStr: string): string;
3983: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
3984: function IsEmptyStr(const S: string; const EmptyChars: TCharSet): Boolean;
3985: { EmptyStr returns true if the given string contains only character from the EmptyChars. }
3986: function ReplaceStr(const S, Srch, Replace: string): string;
3987: { Returns string with every occurrence of Srch string replaced with Replace string. }
3988: function DelSpace(const S: string): string;
3989: { DelSpace return a string with all white spaces removed. }
3990: function DelChars(const S: string; Chr: Char): string;
3991: { DelChars return a string with all Chr characters removed. }
3992: function DelBSpace(const S: string): string;
3993: { DelBSpace trims leading spaces from the given string. }
3994: function DelESpace(const S: string): string;
3995: { DelESpace trims trailing spaces from the given string. }
3996: function DelRSpace(const S: string): string;
3997: { DelRSpace trims leading and trailing spaces from the given string. }
3998: function DelSpace1(const S: string): string;
3999: { DelSpace1 return a string with all non-single white spaces removed. }
4000: function Tab2Space(const S: string; Numb: Byte): string;
4001: { Tab2Space converts any tabulation character in the given string to the Numb spaces characters. }
4002: function NPos(const C: string; S: string; N: Integer): Integer;
4003: { NPos searches for a N-th position of substring C in a given string. }
4004: function MakeStr(C: Char; N: Integer): string;
4005: function MS(C: Char; N: Integer): string;
4006: { MakeStr return a string of length N filled with character C. }
4007: function AddChar(C: Char; const S: string; N: Integer): string;
4008: { AddChar return a string left-padded to length N with characters C. }

```

```

4009: function AddCharR(C: Char; const S: string; N: Integer): string;
4010: { AddCharR return a string right-padded to length N with characters C. }
4011: function LeftStr(const S: string; N: Integer): string;
4012: { LeftStr return a string right-padded to length N with blanks. }
4013: function RightStr(const S: string; N: Integer): string;
4014: { RightStr return a string left-padded to length N with blanks. }
4015: function CenterStr(const S: string; Len: Integer): string;
4016: { CenterStr centers the characters in the string based upon the Len specified. }
4017: function CompStr(const S1, S2: string): Integer;
4018: { CompStr compares S1 to S2, case-sensitivity. return val is -1 if S1 < S2, 0 if S1 = S2, or 1 if S1 > S2. }
4019: function CompText(const S1, S2: string): Integer;
4020: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4021: function Copy2Symb(const S: string; Symb: Char): string;
4022: { Copy2Symb returns a substring of a string S from beginning to first character Symb. }
4023: function Copy2SymbDel(var S: string; Symb: Char): string;
4024: { Copy2SymbDel returns a substr of string S from to first character Symb and removes substring from S. }
4025: function Copy2Space(const S: string): string;
4026: { Copy2Symb returns a substring of a string S from beginning to first white space. }
4027: function Copy2SpaceDel(var S: string): string;
4028: { Copy2SpaceDel returns a substring of a string S from beginning to first
4029:   white space and removes this substring from S. }
4030: function AnsiProperCase(const S: string; const WordDelims: TCharSet): string;
4031: { Returns string, with the first letter of each word in uppercase,
4032:   all other letters in lowercase. Words are delimited by WordDelims. }
4033: function WordCount(const S: string; const WordDelims: TCharSet): Integer;
4034: { WordCount given a set of word delimiters, returns number of words in S. }
4035: function WordPosition(const N: Integer; const S: string; const WordDelims: TCharSet): Integer;
4036: { Given a set of word delimiters, returns start position of N'th word in S. }
4037: function ExtractWord(N: Integer; const S: string; const WordDelims: TCharSet): string;
4038: function ExtractWordPos(N: Integer; const S: string; const WordDelims: TCharSet; var Pos: Integer): string;
4039: function ExtractDelimited(N: Integer; const S: string; const Delims: TCharSet): string;
4040: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4041:   delimiters, return the N'th word in S. }
4042: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TCharSet): string;
4043: { ExtractSubstr given set of word delimiters, returns the substring from S, that started from position Pos. }
4044: function IsWordPresent(const W, S: string; const WordDelims: TCharSet): Boolean;
4045: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4046: function QuotedString(const S: string; Quote: Char): string;
4047: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4048: function ExtractQuotedString(const S: string; Quote: Char): string;
4049: { ExtractQuotedString removes the Quote characters from the beginning and end of a quoted string,
4050:   and reduces pairs of Quote characters within the quoted string to a single character. }
4051: function FindPart(const HelpWilds, InputStr: string): Integer;
4052: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4053: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4054: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4055: function XorString(const Key, Src: ShortString): ShortString;
4056: function XorEncode(const Key, Source: string): string;
4057: function XorDecode(const Key, Source: string): string;
4058: { ** Command line routines ** }
4059: {$IFDEF COMPILER4_UP}
4060: function FindCmdLineSwitch(const Switch: string; SwitchChars: TCharSet; IgnoreCase: Boolean): Boolean;
4061: {$ENDIF}
4062: function GetCmdLineArg(const Switch: string; SwitchChars: TCharSet): string;
4063: { ** Numeric string handling routines ** }
4064: function Numb2USA(const S: string): string;
4065: { Numb2USA converts numeric string S to USA-format. }
4066: function Dec2Hex(N: Longint; A: Byte): string;
4067: function D2H(N: Longint; A: Byte): string;
4068: { Dec2Hex converts the given value to a hexadecimal string representation
4069:   with the minimum number of digits (A) specified. }
4070: function Hex2Dec(const S: string): Longint;
4071: function H2D(const S: string): Longint;
4072: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4073: function Dec2Numb(N: Longint; A, B: Byte): string;
4074: { Dec2Numb converts the given value to a string representation with the
4075:   base equal to B and with the minimum number of digits (A) specified. }
4076: function Numb2Dec(S: string; B: Byte): Longint;
4077: { Numb2Dec converts the given B-based numeric string to the corresponding integer value. }
4078: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4079: { IntToBin converts given value to a bin string representation with min number of digits specified. }
4080: function IntToRoman(Value: Longint): string;
4081: { IntToRoman converts the given value to a roman numeric string representation. }
4082: function RomanToInt(const S: string): Longint;
4083: { RomanToInt converts the given string to an integer value. If the string
4084:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4085: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
4086: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
4087: ***** JvFileUtil *****
4088: procedure CopyFileJ(const FileName, DestName: string; ProgressControl: TControl);
4089: procedure CopyFileEx(const FileName, DestName: string; OverwriteReadOnly, ShellDialog: Boolean; ProgressControl:
  TControl);
4090: procedure MoveFile(const FileName, DestName: TFileName);
4091: procedure MoveFileEx(const FileName, DestName: TFileName; ShellDialog: Boolean);
4092: {$IFDEF COMPILER4_UP}
4093: function GetFileSize(const FileName: string): Int64;
4094: {$ELSE}
4095: function GetFileSize(const FileName: string): Longint;
4096: {$ENDIF}

```

```

4097: function FileDateTime(const FileName: string): TDateTime;
4098: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4099: function DeleteFiles(const FileMask: string): Boolean;
4100: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4101: function ClearDir(const Path: string; Delete: Boolean): Boolean;
4102: function NormalDir(const DirName: string): string;
4103: function RemoveBackSlash(const DirName: string): string;
4104: function ValidFileName(const FileName: string): Boolean;
4105: function DirExists(Name: string): Boolean;
4106: procedure ForceDirectories(Dir: string);
4107: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer;
4108: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4109: {$IFDEF COMPILER4_UP}
4110: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4111: {$ENDIF}
4112: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer;
4113: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4114: {$IFDEF COMPILER4_UP}
4115: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4116: {$ENDIF}
4117: function GetTempDir: string;
4118: function GetWindowsDir: string;
4119: function GetSystemDir: string;
4120: function BrowseDirectory(var AFolderName: string; const DlgText: string; AHelpContext: THelpContext): Boolean;
4121: {$IFDEF WIN32}
4122: function BrowseComputer(var ComputerName: string; const DlgText: string; AHelpContext: THelpContext): Boolean;
4123: function ShortToLongFileName(const ShortName: string): string;
4124: function ShortToLongPath(const ShortName: string): string;
4125: function LongToShortFileName(const LongName: string): string;
4126: function LongToShortPath(const LongName: string): string;
4127: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4128: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4129: {$ENDIF WIN32}
4130: {$IFDEF COMPILER3_UP}
4131: function IsPathDelimiter(const S: string; Index: Integer): Boolean;
4132: {$ENDIF}
4133: function CreateCalculatorForm( AOwner: TComponent; AHelpContext: THelpContext ): TJvCalculatorForm
4134: function CreatePopupCalculator( AOwner: TComponent; ABiDiMode: TBiDiMode ): TWinControl
4135: function CreatePopupCalculator( AOwner: TComponent ): TWinControl
4136: procedure SetupPopupCalculator( PopupCalc: TWinControl; APrecision: Byte; ABeepOnError: Boolean)
4137:
4138: //*****procedure SIRegister_VarHlpr(CL: TPSPascalCompiler);
4139: procedure VariantClear( var V: Variant)
4140: procedure VariantArrayRedim( var V: Variant; High: Integer)
4141: procedure VariantCast( const src: Variant; var dst: Variant; vt: Integer)
4142: procedure VariantCpy( const src: Variant; var dst: Variant)
4143: procedure VariantAdd( const src: Variant; var dst: Variant)
4144: procedure VariantSub( const src: Variant; var dst: Variant)
4145: procedure VariantMul( const src: Variant; var dst: Variant)
4146: procedure VariantDiv( const src: Variant; var dst: Variant)
4147: procedure VariantMod( const src: Variant; var dst: Variant)
4148: procedure VariantAnd( const src: Variant; var dst: Variant)
4149: procedure VariantOr( const src: Variant; var dst: Variant)
4150: procedure VariantXor( const src: Variant; var dst: Variant)
4151: procedure VariantShl( const src: Variant; var dst: Variant)
4152: procedure VariantShr( const src: Variant; var dst: Variant)
4153: function VariantAdd2( const V1: Variant; const V2: Variant ): Variant
4154: function VariantSub2( const V1: Variant; const V2: Variant ): Variant
4155: function VariantMul2( const V1: Variant; const V2: Variant ): Variant
4156: function VariantDiv2( const V1: Variant; const V2: Variant ): Variant
4157: function VariantMod2( const V1: Variant; const V2: Variant ): Variant
4158: function VariantAnd2( const V1: Variant; const V2: Variant ): Variant
4159: function VariantOr2( const V1: Variant; const V2: Variant ): Variant
4160: function VariantXor2( const V1: Variant; const V2: Variant ): Variant
4161: function VariantShl2( const V1: Variant; const V2: Variant ): Variant
4162: function VariantShr2( const V1: Variant; const V2: Variant ): Variant
4163: function VariantNot( const V1: Variant ): Variant
4164: function VariantNeg( const V1: Variant ): Variant
4165: function VariantGetElement( const V: Variant; i1: integer ): Variant;
4166: function VariantGetElement1( const V: Variant; i1, i2: integer ): Variant;
4167: function VariantGetElement2( const V: Variant; i1, i2, i3: integer ): Variant;
4168: function VariantGetElement3( const V: Variant; i1, i2, i3, i4: integer ): Variant;
4169: function VariantGetElement4( const V: Variant; i1, i2, i3, i4, i5: integer ): Variant;
4170: procedure VariantPutElement( var V: Variant; const data: Variant; i1: integer);
4171: procedure VariantPutElement1( var V: Variant; const data: Variant; i1, i2: integer);
4172: procedure VariantPutElement2( var V: Variant; const data: Variant; i1, i2, i3: integer);
4173: procedure VariantPutElement3( var V: Variant; const data: Variant; i1, i2, i3, i4: integer);
4174: procedure VariantPutElement4( var V: Variant; const data: Variant; i1, i2, i3, i4, i5: integer);
4175: end;
4176:
4177: *****unit uPSI_JvgUtils;*****
4178: function IsEven(I: Integer): Boolean;
4179: function InchesToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4180: function CentimetersToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4181: procedure SwapInt(var I1, I2: Integer);
4182: function Spaces(Count: Integer): string;
4183: function DupStr(const Str: string; Count: Integer): string;
4184: function DupChar(C: Char; Count: Integer): string;
4185: procedure Msg(const AMsg: string);

```



```

4186: function RectW(R: TRect): Integer;
4187: function RectH(R: TRect): Integer;
4188: function IncColor(AColor: Longint; AOffset: Byte): Longint;
4189: function DecColor(AColor: Longint; AOffset: Byte): Longint;
4190: function IsItAFilledBitmap(Bmp: TBitmap): Boolean;
4191: procedure DrawTextInRectWithAlign(DC: HDC; R: TRect; const Text: string;
4192:   HAlign: TglHorAlign; VAlign: TglVertAlign; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
4193: procedure DrawTextInRect(DC: HDC; R: TRect; const Text: string; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
4194: procedure ExtTextOutExt(DC: HDC; X, Y: Integer; R: TRect; const Text: string;
4195:   Style: TglTextStyle; ADelinedated, ASupress3D: Boolean; FontColor, DelinColor, HighlightColor, ShadowColor:
  TColor; Illumination: TJvgIllumination; Gradient: TJvgGradient; Font: TFont);
4196: procedure DrawBox(DC: HDC; var R: TRect; Style: TglBoxStyle; BackgrColor: Longint; ATransparent: Boolean);
4197: function DrawBoxEx(DC: HDC; ARect: TRect; Borders: TglSides;
4198:   BevelInner, BevelOuter: TPanelBevel; Bold: Boolean; BackgrColor: Longint; ATransparent: Boolean): TRect;
4199: procedure GradientBox(DC: HDC; R: TRect; Gradient: TJvgGradient; PenStyle, PenWidth: Integer);
4200: procedure ChangeBitmapColor(Bitmap: TBitmap; FromColor, ToColor: TColor);
4201: procedure DrawBitmapExt(DC: HDC; { DC - background & result }
4202:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; {...X,Y _in_ rect}
4203:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4204:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4205: procedure CreateBitmapExt(DC: HDC; { DC - background & result }
4206:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; {...X,Y _in_ rect}
4207:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4208:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4209: procedure BringParentWindowToTop(Wnd: TWinControl);
4210: function GetParentForm(Control: TControl): TForm;
4211: procedure GetWindowImageFrom(Control: TWinControl; X, Y: Integer; ADrawSelf, ADrawChildWindows: Boolean; DC: HDC)
4212: procedure GetWindowImage(Control: TWinControl; ADrawSelf, ADrawChildWindows: Boolean; DC: HDC);
4213: procedure GetParentImageRect(Control: TControl; Rect: TRect; DC: HDC);
4214: function CreateRotatedFont(F: TFont; Escapement: Integer): HFONT;
4215: function FindMainWindow(const AWndClass, AWndTitle: string): THandle;
4216: procedure CalcShadowAndHighlightColors(BaseColor: TColor; Colors: TJvgLabelColors);
4217: function CalcMathString(AExpression: string): Single;
4218: function IIF(AExpression: Boolean; IfTrue, IfFalse: Variant): Variant; overload;
4219: function IIF(AExpression: Boolean; const IfTrue, IfFalse: string): string; overload;
4220: function GetTransparentColor(Bitmap: TBitmap; AutoTrColor: TglAutoTransparentColor): TColor;
4221: procedure TypeStringOnKeyboard(const S: string);
4222: function NextStringGridCell( Grid: TStringGrid ) : Boolean;
4223: procedure DrawTextExtAligned(Canvas: TCanvas; const
  Text: string; R: TRect; Alignment: TglAlignment; WordWrap: Boolean);
4224: procedure LoadComponentFromTextFile(Component: TComponent; const FileName: string);
4225: procedure SaveComponentToTextFile(Component: TComponent; const FileName: string);
4226: function ComponentToString(Component: TComponent): string;
4227: procedure StringToComponent(Component: TComponent; const Value: string);
4228: function PlayWaveResource(const ResName: string): Boolean;
4229: function UserName: string;
4230: function ComputerName: string;
4231: function CreateIniFileName: string;
4232: function ExpandString(const Str: string; Len: Integer): string;
4233: function Transliterate(const Str: string; RusToLat: Boolean): string;
4234: function IsSmallFonts: Boolean;
4235: function SystemColorDepth: Integer;
4236: function GetFileTypeJ(const FileName: string): TglFileType;
4237: function GetFileType( hFile : THandle ) : DWORD';
4238: function FindControlAtPt(Control: TWinControl; Pt: TPoint; MinClass: TClass): TControl;
4239: function StrPosExt(const Str1, Str2: PChar; Str2Len: DWORD): PChar;
4240:
4241: { *****Utility routines of unit classes}
4242: function LineStart(Buffer, BufPos: PChar): PChar
4243: function ExtractStrings(Separators, WhiteSpace: TSysCharSet; Content: PChar; '+
4244:   'Strings: TStrings): Integer
4245: function TestStreamFormat( Stream : TStream ) : TStreamOriginalFormat
4246: procedure RegisterClass( AClass : TPersistentClass)
4247: procedure RegisterClasses( AClasses : array of TPersistentClass)
4248: procedure RegisterClassAlias( AClass : TPersistentClass; const Alias : string)
4249: procedure UnRegisterClass( AClass : TPersistentClass)
4250: procedure UnRegisterClasses( AClasses : array of TPersistentClass)
4251: procedure UnRegisterModuleClasses( Module : HMODULE)
4252: function FindGlobalComponent( const Name : string ) : TComponent
4253: function IsUniqueGlobalComponentName( const Name : string ) : Boolean
4254: function InitInheritedComponent( Instance : TComponent; RootAncestor : TClass ) : Boolean
4255: function InitComponentRes( const ResName : string; Instance : TComponent ) : Boolean
4256: function ReadComponentRes( const ResName : string; Instance : TComponent ) : TComponent
4257: function ReadComponentResEx( HInstance : THandle; const ResName : string ) : TComponent
4258: function ReadComponentResFile( const FileName : string; Instance : TComponent ) : TComponent
4259: procedure WriteComponentResFile( const FileName : string; Instance : TComponent)
4260: procedure GlobalFixupReferences
4261: procedure GetFixupReferenceNames( Root : TComponent; Names : TStrings)
4262: procedure GetFixupInstanceNames(Root: TComponent; const ReferenceRootName string; Names : TStrings)
4263: procedure RedirectFixupReferences( Root : TComponent; const OldRootName, NewRootName : string)
4264: procedure RemoveFixupReferences( Root : TComponent; const RootName : string)
4265: procedure RemoveFixups( Instance : TPersistent)
4266: function FindNestedComponent( Root : TComponent; const NamePath : string ) : TComponent
4267: procedure BeginGlobalLoading
4268: procedure NotifyGlobalLoading
4269: procedure EndGlobalLoading
4270: function GetUltimateOwner1( ACollection : TCollection ) : TPersistent;
4271: function GetUltimateOwner( APersistent : TPersistent ) : TPersistent;
4272: // AddTypes('TWndMethod', 'Procedure (var Message : TMessage)

```

```

4273: //Function MakeObjectInstance( Method : TWndMethod) : Pointer
4274: Procedure FreeObjectInstance( ObjectInstance : Pointer)
4275: // Function AllocateHWnd( Method : TWndMethod) : HWND
4276: Procedure DeallocateHWnd( Wnd : HWND)
4277: Function AncestorIsValid( Ancestor : TPersistent; Root, RootAncestor : TComponent) : Boolean
4278: *****unit uPSI_SqlTimSt and DB;*****
4279: Procedure VarSQLTimeStampCreate4( var aDest : Variant; const ASQLTimeStamp : TSQLTimeStamp);
4280: Function VarSQLTimeStampCreate3: Variant;
4281: Function VarSQLTimeStampCreate2( const AValue : string) : Variant;
4282: Function VarSQLTimeStampCreate1( const AValue : TDateTime) : Variant;
4283: Function VarSQLTimeStampCreate( const ASQLTimeStamp : TSQLTimeStamp) : Variant;
4284: Function VarSQLTimeStamp : TVarType
4285: Function VarIsSQLTimeStamp( const aValue : Variant) : Boolean;
4286: Function LocalToUTC( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp //beta
4287: Function UTCToLocal( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp //beta
4288: Function VarToSQLTimeStamp( const aValue : Variant) : TSQLTimeStamp
4289: Function SQLTimeStampToStr( const Format : string; DateTime : TSQLTimeStamp) : string
4290: Function SQLDayOfWeek( const DateTime : TSQLTimeStamp) : integer
4291: Function DateTimeToSQLTimeStamp( const DateTime : TDateTime) : TSQLTimeStamp
4292: Function SQLTimeStampToDateTime( const DateTime : TSQLTimeStamp) : TDateTime
4293: Function TryStrToSQLTimeStamp( const S : string; var TimeStamp : TSQLTimeStamp) : Boolean
4294: Function StrToSQLTimeStamp( const S : string) : TSQLTimeStamp
4295: Procedure CheckSqlTimeStamp( const ASQLTimeStamp : TSQLTimeStamp)
4296: Function ExtractFieldName( const Fields : string; var Pos : Integer) : string;
4297: Function ExtractFieldName( const Fields : WideString; var Pos : Integer) : WideString;
4298: //Procedure RegisterFields( const FieldClasses : array of TFieldClass)
4299: Procedure DatabaseError( const Message : WideString; Component : TComponent)
4300: Procedure DatabaseErrorFmt( const Message:WideString; const Args:array of const;Component:TComponent)
4301: Procedure DisposeMem( var Buffer, Size : Integer)
4302: Function BuffersEqual( Buf1, Buf2 : Pointer; Size : Integer) : Boolean
4303: Function GetFieldProperty(DataSet:TDataSet; Control:TComponent; const FieldName: WideString): TField
4304: Function VarTypeToDataType( VarType : Integer) : TFieldType
4305: *****unit JvStrings;*****
4306: {template functions}
4307: function ReplaceFirst(const SourceStr, FindStr, ReplaceStr: string): string;
4308: function ReplaceLast(const SourceStr, FindStr, ReplaceStr: string): string;
4309: function InsertLastBlock(var SourceStr: string; BlockStr: string): Boolean;
4310: function RemoveMasterBlocks(const SourceStr: string): string;
4311: function RemoveFields(const SourceStr: string): string;
4312: {http functions}
4313: function URLEncode(const Value: AnsiString): AnsiString; // Converts string To A URLEncoded string
4314: function URLDecode(const Value: AnsiString): AnsiString; // Converts string From A URLEncoded string
4315: {set functions}
4316: procedure SplitSet(AText: string; AList: TStringList);
4317: function JoinSet(AList: TStringList): string;
4318: function FirstOfSet(const AText: string): string;
4319: function LastOfSet(const AText: string): string;
4320: function CountOfSet(const AText: string): Integer;
4321: function SetRotateRight(const AText: string): string;
4322: function SetRotateLeft(const AText: string): string;
4323: function SetPick(const AText: string; AIndex: Integer): string;
4324: function SetSort(const AText: string): string;
4325: function SetUnion(const Set1, Set2: string): string;
4326: function SetIntersect(const Set1, Set2: string): string;
4327: function SetExclude(const Set1, Set2: string): string;
4328: {replace any <,> etc by &lt; &gt;}
4329: function XMLSafe(const AText: string): string;
4330: {simple hash, Result can be used in Encrypt}
4331: function Hash(const AText: string): Integer;
4332: { Base64 encode and decode a string }
4333: function B64Encode(const S: AnsiString): AnsiString;
4334: function B64Decode(const S: AnsiString): AnsiString;
4335: {Basic encryption from a Borland Example}
4336: function Encrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4337: function Decrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4338: {Using Encrypt and Decrypt in combination with B64Encode and B64Decode}
4339: function EncryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4340: function DecryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4341: procedure CSVToTags(Src, Dst: TStringList);
4342: // converts a csv list to a tagged string list
4343: procedure TagsToCSV(Src, Dst: TStringList);
4344: // converts a tagged string list to a csv list
4345: // only fieldnames from the first record are scanned ib the other records
4346: procedure ListSelect(Src, Dst: TStringList; const AKey, AValue: string);
4347: {selects akey=avalue from Src and returns recordset in Dst}
4348: procedure ListFilter(Src: TStringList; const AKey, AValue: string);
4349: {filters Src for akey=avalue}
4350: procedure ListOrderBy(Src: TStringList; const AKey: string; Numeric: Boolean);
4351: {orders a tagged Src list by akey}
4352: function PosStr(const FindString, SourceString: string;
4353:   StartPos: Integer = 1): Integer;
4354: { PosStr searches the first occurrence of a substring FindString in a string
4355:   given by SourceString with case sensitivity (upper and lower case characters
4356:   are differed). This function returns the index value of the first character
4357:   of a specified substring from which it occurs in a given string starting with
4358:   StartPos character index. If a specified substring is not found Q_PosStr
4359:   returns zero. author of algorithm is Peter Morris (UK) (Faststrings unit from www.torry.ru). }
4360: function PosStrLast(const FindString, SourceString: string): Integer;
4361: {finds the last occurrence}

```

```

4362: function LastPosChar(const FindChar: Char; SourceString: string): Integer;
4363: function PosText(const FindString, SourceString: string; StartPos: Integer = 1): Integer;
4364: { PosText searches the first occurrence of a substring FindString in a string
4365:   given by SourceString without case sensitivity (upper and lower case characters are not differed). This
   function returns the index value of the first character of a specified substring from which it occurs in a
   given string starting with Start
4366: function PosTextLast(const FindString, SourceString: string): Integer;
4367: {finds the last occurrence}
4368: function NameValuesToXML(const AText: string): string;
4369: {$IFDEF MSWINDOWS}
4370: procedure LoadResourceFile(AFile: string; MemStream: TMemoryStream);
4371: {$ENDIF MSWINDOWS}
4372: procedure DirFiles(const ADir, AMask: string; AFileList: TStringList);
4373: procedure RecurseDirFiles(const ADir: string; var AFileList: TStringList);
4374: procedure RecurseDirProgs(const ADir: string; var AFileList: TStringList);
4375: procedure SaveString(const AFile, AText: string);
4376: procedure SaveStringasFile(const AFile, AText: string);
4377: function LoadStringJ(const AFile: string): string;
4378: function LoadStringOfFile(const AFile: string): string;
4379: procedure SaveStringToFile(const AFile, AText: string);
4380: function LoadStringFromFile(const AFile: string): string;
4381: function HexToColor(const AText: string): TColor;
4382: function UppercaseHTMLTags(const AText: string): string;
4383: function LowercaseHTMLTags(const AText: string): string;
4384: procedure GetHTMLAnchors(const AFile: string; AList: TStringList);
4385: function RelativePath(const ASrc, ADst: string): string;
4386: function GetToken(var Start: Integer; const SourceText: string): string;
4387: function PosNonSpace(Start: Integer; const SourceText: string): Integer;
4388: function PosEscaped(Start: Integer; const SourceText, FindText: string; EscapeChar: Char): Integer;
4389: function DeleteEscaped(const SourceText: string; EscapeChar: Char): string;
4390: function BeginOfAttribute(Start: Integer; const SourceText: string): Integer;
4391: // parses the beginning of an attribute: space + alpha character
4392: function ParseAttribute(var Start: Integer; const SourceText: string; var AName, AValue: string): Boolean;
4393: //parses a name="value" attrib from Start; returns 0 when not found or else the position behind attribute
4394: procedure ParseAttributes(const SourceText: string; Attributes: TStringList);
4395: // parses all name=value attributes to the attributes TStringList
4396: function HasStrValue(const AText, AName: string; var AValue: string): Boolean;
4397: // checks if a name="value" pair exists and returns any value
4398: function GetStrValue(const AText, AName, ADefault: string): string;
4399: // retrieves string value from a line like:
4400: // name="jan verhoeven" email="jan1 dott verhoeven att wxs dott nl"
4401: // returns ADefault when not found
4402: function GetHTMLColorValue(const AText, AName: string; ADefault: TColor): TColor;
4403: // same for a color
4404: function GetIntValue(const AText, AName: string; ADefault: Integer): Integer;
4405: // same for an Integer
4406: function GetFloatValue(const AText, AName: string; ADefault: Extended): Extended;
4407: // same for a float
4408: function GetBoolValue(const AText, AName: string): Boolean;
4409: // same for Boolean but without default
4410: function GetValue(const AText, AName: string): string;
4411: //retrieves string value from a line like: name="jan verhoeven" email="jan1 verhoeven att wxs dott nl"
4412: procedure SetValue(var AText: string; const AName, AValue: string);
4413: // sets a string value in a line
4414: procedure DeleteValue(var AText: string; const AName: string);
4415: // deletes a AName="value" pair from AText
4416: procedure GetNames(AText: string; AList: TStringList);
4417: // get a list of names from a string with name="value" pairs
4418: function GetHTMLColor(AColor: TColor): string;
4419: // converts a color value to the HTML hex value
4420: function BackPosStr(Start: Integer; const FindString, SourceString: string): Integer;
4421: // finds a string backward case sensitive
4422: function BackPosText(Start: Integer; const FindString, SourceString: string): Integer;
4423: // finds a string backward case insensitive
4424: function PosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4425:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4426: // finds a text range, e.g. <TD>...</TD> case sensitive
4427: function PosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4428:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4429: // finds a text range, e.g. <TD>...</td> case insensitive
4430: function BackPosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4431:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4432: // finds a text range backward, e.g. <TD>...</TD> case sensitive
4433: function BackPosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4434:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4435: // finds a text range backward, e.g. <TD>...</td> case insensitive
4436: function PosTag(Start: Integer; SourceString: string; var RangeBegin: Integer;
4437:   var RangeEnd: Integer): Boolean;
4438: // finds a HTML or XML tag: <....>
4439: function InnerTag(Start: Integer; const HeadString, TailString, SourceString: string;
4440:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4441: // finds the innertext between opening and closing tags
4442: function Easter(NYear: Integer): TDateTime;
4443: // returns the easter date of a year.
4444: function GetWeekNumber(Today: TDateTime): string;
4445: //gets a datecode. Returns year and weeknumber in format: YYWW
4446: function ParseNumber(const S: string): Integer;
4447: // parse number returns the last position, starting from 1
4448: function ParseDate(const S: string): Integer;

```

```

4449: // parse a SQL style data string from positions 1,
4450: // starts and ends with #
4451:
4452: *****unit JvJCLUtils;*****
4453:
4454: function VarIsInt(Value: Variant): Boolean;
4455: // VarIsInt returns VarIsOrdinal-[varBoolean]
4456: { PosIdx returns the index of the first appearance of SubStr in Str. The search starts at index "Index". }
4457: function PosIdx(const SubStr, S: string; Index: Integer = 0): Integer;
4458: function PosIdxW(const SubStr, S: WideString; Index: Integer = 0): Integer;
4459: function PosLastCharIdx(Ch: Char; const S: string; Index: Integer = 0): Integer;
4460: { GetWordOnPos returns Word from string, S, on the cursor position, P }
4461: function GetWordOnPos(const S: string; const P: Integer): string;
4462: function GetWordOnPosW(const S: WideString; const P: Integer): WideString;
4463: function GetWordOnPos2(const S: string; P: Integer; var iBeg, iEnd: Integer): string;
4464: function GetWordOnPos2W(const S: WideString; P: Integer; var iBeg, iEnd: Integer): WideString;
4465: { GetWordOnPosEx working like GetWordOnPos function, but
4466:   also returns Word position in iBeg, iEnd variables }
4467: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4468: function GetWordOnPosExW(const S: WideString; const P: Integer; var iBeg, iEnd: Integer): WideString;
4469: function GetNextWordPosEx(const Text: string; StartIndex: Integer; var iBeg, iEnd: Integer): string;
4470: function GetNextWordPosExW(const Text: WideString; StartIndex: Integer; var iBeg, iEnd: Integer): WideString;
4471: procedure GetEndPosCaret(const Text: string; CaretX, CaretY: Integer; var X, Y: Integer);
4472: { GetEndPosCaret returns the caret position of the last char. For the position
4473:   after the last char of Text you must add 1 to the returned X value. }
4474: procedure GetEndPosCaretW(const Text: WideString; CaretX, CaretY: Integer; var X, Y: Integer);
4475: { GetEndPosCaret returns the caret position of the last char. For the position
4476:   after the last char of Text you must add 1 to the returned X value. }
4477: { SubStrBySeparator returns substring from string, S, separated with Separator string }
4478: function SubStrBySeparator(const S: string; const Index: Integer; const
  Separator: string; StartIndex: Integer = 1): string;
4479: function SubStrBySeparatorW(const S: WideString; const Index: Integer; const
  Separator: WideString; StartIndex: Integer = 1): WideString;
4480: { SubStrEnd same to previous function but Index numerated from the end of string }
4481: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4482: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
4483: function SubWord(P: PChar; var P2: PChar): string;
4484: function CurrencyByWord(Value: Currency): string;
4485: { GetLineByPos returns the Line number, there the symbol Pos is pointed. Lines separated with #13 symbol }
4486: function GetLineByPos(const S: string; const Pos: Integer): Integer;
4487: { GetXYByPos is same as GetLineByPos, but returns X position in line as well }
4488: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4489: procedure GetXYByPosW(const S: WideString; const Pos: Integer; var X, Y: Integer);
4490: { ReplaceString searches for all substrings, OldPattern,
4491:   in a string, S, and replaces them with NewPattern }
4492: function ReplaceString(S: string; const OldPattern, NewPattern: string; StartIndex: Integer = 1): string;
4493: function ReplaceStringW(S: WideString; const OldPattern, NewPattern:
  WideString; StartIndex: Integer = 1): WideString;
4494: { ConcatSep concatenate S1 and S2 strings with Separator. if S = '' then separator not included }
4495: function ConcatSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
  SUPPORTS_INLINE}
4496: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
4497: function ConcatLeftSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
  SUPPORTS_INLINE}
4498:
4499: { Next 4 function for russian chars transliterating.
4500:   This functions are needed because Oem2Ansi and AnsiOem functions sometimes suck }
4501: procedure Dos2Win(var S: AnsiString);
4502: procedure Win2Dos(var S: AnsiString);
4503: function Dos2WinRes(const S: AnsiString): AnsiString; inline; {$IFDEF SUPPORTS_INLINE}
4504: function Win2DosRes(const S: AnsiString): AnsiString; inline; {$IFDEF SUPPORTS_INLINE}
4505: function Win2Koi(const S: AnsiString): AnsiString;
4506: { FillString fills the string Buffer with Count Chars }
4507: procedure FillString(var Buffer: string; Count: Integer; const Value: Char); overload;
4508: procedure FillString(var Buffer: string; StartIndex, Count: Integer; const Value: Char); overload;
4509: { MoveString copies Count Chars from Source to Dest }
4510: procedure MoveString(const Source: string; var Dest: string; Count: Integer); {$IFDEF SUPPORTS_INLINE}
  inline; {$ENDIF SUPPORTS_INLINE} overload;
4511: procedure MoveString(const Source: string; SrcStartIdx: Integer; var Dest: string;
  DstStartIdx: Integer; Count: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE} overload;
4512: { FillWideChar fills Buffer with Count WideChars (2 Bytes) }
4513: procedure FillWideChar(var Buffer: string; Count: Integer; const Value: WideChar);
4514: { MoveWideChar copies Count WideChars from Source to Dest }
4515: procedure MoveWideChar(const Source: string; var Dest: string; Count: Integer); {$IFDEF SUPPORTS_INLINE}
  inline; {$ENDIF SUPPORTS_INLINE}
4516: { FillNativeChar fills Buffer with Count NativeChars }
4517: procedure FillNativeChar(var Buffer: string; Count: Integer; const Value: Char); // D2009 internal error {$IFDEF
  SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4518: { MoveNativeChar copies Count WideChars from Source to Dest }
4519: procedure MoveNativeChar(const Source: string; var Dest: string; Count: Integer); // D2009 internal error {$IFDEF
  SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4520: { IsSubString() compares the sub string to the string. Indices are 1th based. }
4521: function IsSubString(const S: string; StartIndex: Integer; const SubStr: string): Boolean;
4522: { Spaces returns string consists on N space chars }
4523: function Spaces(const N: Integer): string;
4524: { AddSpaces adds spaces to string S, if its Length is smaller than N }
4525: function AddSpaces(const S: string; const N: Integer): string;
4526: function SpacesW(const N: Integer): WideString;
4527: function AddSpacesW(const S: WideString; const N: Integer): WideString;
4528:

```



```

4529: { function LastDateRUS for russian users only }
4530: { returns date relative to current date: 'âââ âîý îâçââ' }
4531: function LastDateRUS(const Dat: TDateTime): string;
4532: { CurrencyToStr format Currency, Cur, using ffCurrency float format }
4533: function CurrencyToStr(const Cur: Currency): string;
4534: { HasChar returns True, if Char, Ch, contains in string, S }
4535: function HasChar(const Ch: Char; const S: string): Boolean;
4536: function HasCharW(const Ch: WideChar; const S: WideString): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4537: function HasAnyChar(const Chars: string; const S: string): Boolean;
4538: {$IFDEF COMPILER12_UP}
4539: function CharInSet(const Ch: AnsiChar; const SetOfChar: TSysCharSet): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4540: {$ENDIF ~COMPILER12_UP}
4541: function CharInSetW(const Ch: WideChar; const SetOfChar: TSysCharSet): Boolean; inline; {$ENDIF
SUPPORTS_INLINE}
4542: function CountOfChar(const Ch: Char; const S: string): Integer;
4543: function DefStr(const S: string; Default: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
SUPPORTS_INLINE}
4544: { StrLICompW2 is a faster replacement for JclUnicode.StrLICompW }
4545: function StrLICompW2(S1, S2: PWideChar; MaxLen: Integer): Integer;
4546: function StrPosW(S, SubStr: PWideChar): PWideChar;
4547: function StrLenW(S: PWideChar): Integer;
4548: function TrimW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4549: function TrimLeftW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
SUPPORTS_INLINE}
4550: function TrimRightW(const S: WideString): WideString; inline; {$ENDIF SUPPORTS_INLINE}
4551: TPixelFormat, '( pfDevice, pf1bit, pf4bit, pf8bit, pf24bit )
4552: TMappingMethod, '( mmHistogram, mmQuantize, mmTrunc784, mmTrunc666, mmTripel, mmGrayscale )
4553: function GetBitmapPixelFormat( Bitmap : TBitmap ) : TPixelFormat
4554: function GetPaletteBitmapFormat( Bitmap : TBitmap ) : TPixelFormat
4555: procedure SetBitmapPixelFormat( Bitmap: TBitmap; PixelFormat: TPixelFormat; Method: TMappingMethod)
4556: function BitmapToMemoryStream(Bitmap: TBitmap; PixelFormat: TPixelFormat; Method: TMappingMethod): TMemoryStream;
4557: procedure GrayscaleBitmap( Bitmap : TBitmap)
4558: function BitmapToMemory( Bitmap : TBitmap; Colors : Integer ) : TStream
4559: procedure SaveBitmapToFile( const Filename : string; Bitmap : TBitmap; Colors : Integer)
4560: function ScreenPixelFormat : TPixelFormat
4561: function ScreenColorCount : Integer
4562: procedure TileImage( Canvas : TCanvas; Rect : TRect; Image : TGraphic)
4563: function ZoomImage( ImageW, ImageH, MaxW, MaxH : Integer; Stretch : Boolean ) : TPoint
4564: // SIRegister_TJvGradient(CL);
4565:
4566: {***** files routines}
4567: procedure SetDelimitedText(List: TStrings; const Text: string; Delimiter: Char);
4568: const
4569: {$IFDEF MSWINDOWS}
4570: DefaultCaseSensitivity = False;
4571: {$ENDIF MSWINDOWS}
4572: {$IFDEF UNIX}
4573: DefaultCaseSensitivity = True;
4574: {$ENDIF UNIX}
4575: { GenTempFileName returns temporary file name on
4576: drive, there FileName is placed }
4577: function GenTempFileName(FileName: string): string;
4578: { GenTempFileNameExt same to previous function, but
4579: returning filename has given extension, FileExt }
4580: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
4581: { ClearDir clears folder Dir }
4582: function ClearDir(const Dir: string): Boolean;
4583: { DeleteDir clears and than delete folder Dir }
4584: function DeleteDir(const Dir: string): Boolean;
4585: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4586: function FileEquMask(FileName, Mask: TFileName; CaseSensitive: Boolean=DefaultCaseSensitivity): Boolean;
4587: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4588: Masks must be separated with SepPath (MSW: ';' / UNIX: ':') }
4589: function FileEquMasks(FileName, Masks: TFileName;
4590: CaseSensitive: Boolean = DefaultCaseSensitivity): Boolean;
4591: function DeleteFiles(const Folder: TFileName; const Masks: string): Boolean;
4592: {$IFDEF MSWINDOWS}
4593: { LZFileExpand expand file, FileSource,
4594: into FileDest. Given file must be compressed, using MS Compress program }
4595: function LZFileExpand(const FileSource, FileDest: string): Boolean;
4596: {$ENDIF MSWINDOWS}
4597: { FileGetInfo fills SearchRec record for specified file attributes}
4598: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
4599: { HasSubFolder returns True, if folder APath contains other folders }
4600: function HasSubFolder(APath: TFileName): Boolean;
4601: { IsEmptyFolder returns True, if there are no files or
4602: folders in given folder, APath}
4603: function IsEmptyFolder(APath: TFileName): Boolean;
4604: { AddSlash returns string with added slash Char to Dir parameter, if needed }
4605: function AddSlash(const Dir: TFileName): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4606: { AddPath returns FileName with Path, if FileName not contain any path }
4607: function AddPath(const FileName, Path: TFileName): TFileName;
4608: function AddPaths(const PathList, Path: string): string;
4609: function ParentPath(const Path: TFileName): TFileName;
4610: function FindInPath(const FileName, PathList: string): TFileName;
4611: { DeleteReadOnlyFile clears R/O file attribute and delete file }
4612: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
4613: { HasParam returns True, if program running with specified parameter, Param }
4614: function HasParam(const Param: string): Boolean;

```

```

4615: function HasSwitch(const Param: string): Boolean;
4616: function Switch(const Param: string): string;
4617: { ExePath returns ExtractFilePath(ParamStr(0)) }
4618: function ExePath: TFileName; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4619: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
4620: //function FileTimeToDateTime(const FT: TFileTime): TDateTime;
4621: procedure FileTimeToDosDateTimeDword(const FT: TFileTime; out Dft: DWORD);
4622: function MakeValidFileName(const FileName: TFileName; ReplaceBadChar: Char): TFileName;
4623: {**** Graphic routines }
4624: { IstTFontSelected returns True, if True Type font is selected in specified device context }
4625: function IstTFontSelected(const DC: HDC): Boolean;
4626: function KeyPressed(VK: Integer): Boolean;
4627: function isKeyPressed: boolean; //true if key on memo2 (shell output) is pressed
4628: { TrueInflateRect inflates rect in other method, than InflateRect API function }
4629: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
4630: {**** Color routines }
4631: procedure RGBToHSV(R, G, B: Integer; var H, S, V: Integer);
4632: function RGBToBGR(Value: Cardinal): Cardinal;
4633: //function ColorToPrettyName(Value: TColor): string;
4634: //function PrettyNameToColor(const Value: string): TColor;
4635: {**** other routines }
4636: procedure SwapInt(var Int1, Int2: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4637: function IntPower(Base, Exponent: Integer): Integer;
4638: function ChangeTopException(E: TObject): TObject; // Linux version writes error message to ErrOutput
4639: function StrToBool(const S: string): Boolean;
4640: function Var2Type(V: Variant; const DestVarType: Integer): Variant;
4641: function VarToInt(V: Variant): Integer;
4642: function VarToFloat(V: Variant): Double;
4643: { following functions are not documented because they not work properly sometimes, so do not use them }
4644: // (rom) ReplaceStrings1, GetSubStr removed
4645: function GetLongFileName(const FileName: string): string;
4646: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
4647: function GetParameter: string;
4648: function GetComputerID: string;
4649: function GetComputerName: string;
4650: {**** string routines }
4651: { ReplaceAllStrings searches for all substrings, Words,
4652:   in a string, S, and replaces them with Phrases with the same Index. }
4653: function ReplaceAllStrings(const S: string; Words, Phrases: TStrings): string;
4654: { ReplaceStrings searches the Word in a string, S, on PosBeg position,
4655:   in the list, Words, and if founds, replaces this Word with string from another list, Phrases, with the
   same Index, and then update NewSelStart variable }
4656: function ReplaceStrings(const S: string; PosBeg, Len: Integer; Words, Phrases: TStrings; var NewSelStart: Integer): string;
4657: { CountOfLines calculates the lines count in a string, S,
4658:   each line must be separated from another with CrLf sequence }
4659: function CountOfLines(const S: string): Integer;
4660: { DeleteLines deletes all lines from strings which in the words, words.
4661:   The word of will be deleted from strings. }
4662: procedure DeleteOfLines(Ss: TStrings; const Words: array of string);
4663: { DeleteEmptyLines deletes all empty lines from strings, Ss.
4664:   Lines contained only spaces also deletes. }
4665: procedure DeleteEmptyLines(Ss: TStrings);
4666: { SQLAddWhere addes or modifies existing where-statement, where,
4667:   to the strings, SQL. Note: If strings SQL already contains where-statement,
4668:   it must be started on the begining of any line }
4669: procedure SQLAddWhere(SQL: TStrings; const Where: string);
4670: {**** files routines - }
4671: {$IFDEF MSWINDOWS}
4672: { ResSaveToFile save resource named as Name with Typ type into file FileName.
4673:   Resource can be compressed using MS Compress program }
4674: function ResSaveToFile(const Typ, Name: string; const Compressed: Boolean; const FileName: string): Boolean;
4675: function ResSaveToFileEx(Instance: HINST; Typ, Name: PChar; const Compressed: Boolean; const FileName: string):
   Boolean;
4676: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
4677: {$ENDIF MSWINDOWS}
4678: { IniReadSection read section, Section, from ini-file,
4679:   IniFileName, into strings, Ss. This function reads ALL strings from specified section.
4680:   Note: TIniFile.ReadSection function reads only strings with '=' symbol. }
4681: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
4682: { LoadTextFile load text file, FileName, into string }
4683: function LoadTextFile(const FileName: TFileName): string;
4684: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4685: { ReadFolder reads files list from disk folder, Folder, that are equal to mask, Mask, into strings, FileList }
4686: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
4687: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
4688: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
4689: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
4690: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
4691: function RATextOutEx(Canvas: TCanvas; const R, RClip: TRect; const S: string; const CalcHeight: Boolean): Integer;
4692: { RATextCalcHeight calculate needed height for
4693:   correct output, using RATextOut or RATextOutEx functions }
4694: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
4695: { Cinema draws some visual effect }
4696: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
4697: { Roughed fills rect with special 3D pattern }
4698: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
4699: { BitmapFromBitmap creates new small bitmap from part of source bitmap, SrcBitmap, with specified width
   and height, AWidth, AHeight and placed on a specified Index, Index in the source bitmap }
4700: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;

```

```

4701: { TextWidth calculate text with for writing using standard desktop font }
4702: function TextWidth(const AStr: string): Integer;
4703: { TextHeight calculate text height for writing using standard desktop font }
4704: function TextHeight(const AStr: string): Integer;
4705: procedure SetChildPropOrd(Owner: TComponent; const PropName: string; Value: Longint);
4706: procedure Error(const Msg: string);
4707: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
4708:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
4709: {example Text parameter: 'Item 1<b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green<c:blue>blue</i>' }
4710: function ItemHtDraw(Canvas: TCanvas; Rect: TRect;
4711:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): string;
4712: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;
4713:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): Integer;
4714: function ItemHtPlain(const Text: string): string;
4715: { ClearList - clears list of TObject }
4716: procedure ClearList(List: TList);
4717: procedure MemStreamToClipboard(MemStream: TMemoryStream; const Format: Word);
4718: procedure ClipboardToMemStream(MemStream: TMemoryStream; const Format: Word);
4719: { RTTI support }
4720: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
4721: function GetPropStr(Obj: TObject; const PropName: string): string;
4722: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
4723: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
4724: procedure PrepareIniSection(Ss: TStringList);
4725: { following functions are not documented because they are don't work properly, so don't use them }
4726: // (rom) from JvBandWindows to make it obsolete
4727: function PointL(const X, Y: Longint): TPointL; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4728: // (rom) from JvBandUtils to make it obsolete
4729: function iif(const Test: Boolean; const ATrue, AFalse: Variant): Variant;
4730: procedure CopyIconToClipboard(Icon: TIcon; BackColor: TColor);
4731: function CreateIconFromClipboard: TIcon;
4732: { begin JvIconClipboardUtils } { Icon clipboard routines }
4733: function CF_ICON: Word;
4734: procedure AssignClipboardIcon(Icon: TIcon);
4735: { Real-size icons support routines (32-bit only) }
4736: procedure GetIconSize(Icon: HICON; var W, H: Integer);
4737: function CreateRealSizeIcon(Icon: TIcon): HICON;
4738: procedure DrawRealSizeIcon(Canvas: TCanvas; Icon: TIcon; X, Y: Integer);
4739: {end JvIconClipboardUtils }
4740: function CreateScreenCompatibleDC: HDC;
4741: function InvalidateRect(hWnd: HWND; const lpRect: TRect; bErase: BOOL): BOOL; overload; {$IFDEF
SUPPORTS_INLINE} inline; {$ENDIF}
4742: function InvalidateRect(hWnd: HWND; lpRect: PRect; bErase: BOOL): BOOL; overload;
4743: { begin JvRLE } // (rom) changed API for inclusion in JCL
4744: procedure RleCompressTo(InStream, OutStream: TStream);
4745: procedure RleDecompressTo(InStream, OutStream: TStream);
4746: procedure RleCompress(Stream: TStream);
4747: procedure RleDecompress(Stream: TStream);
4748: { end JvRLE } { begin JvDateUtil }
4749: function CurrentYear: Word; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4750: function IsLeapYear(AYear: Integer): Boolean;
4751: function DaysInAMonth(const AYear, AMonth: Word): Word;
4752: function DaysPerMonth(AYear, AMonth: Integer): Integer;
4753: function FirstDayOfPrevMonth: TDateTime;
4754: function LastDayOfPrevMonth: TDateTime;
4755: function FirstDayOfNextMonth: TDateTime;
4756: function ExtractDay(ADate: TDateTime): Word;
4757: function ExtractMonth(ADate: TDateTime): Word;
4758: function ExtractYear(ADate: TDateTime): Word;
4759: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
4760: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime; inline; {$ENDIF SUPPORTS_INLINE}
4761: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
4762: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
4763: function ValidDate(ADate: TDateTime): Boolean;
4764: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
4765: function MonthsBetween(Date1, Date2: TDateTime): Double;
4766: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
4767: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
4768: function DaysBetween(Date1, Date2: TDateTime): Longint;
4769: { The same as previous but if Date2 < Date1 result = 0 }
4770: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
4771: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
4772: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
4773: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
4774: function IncMsec(ATime: TDateTime; Delta: Integer): TDateTime;
4775: function CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
4776: { String to date conversions }
4777: function GetDateOrder(const DateFormat: string): TDateOrder;
4778: function MonthFromName(const S: string; MaxLen: Byte): Byte;
4779: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
4780: function StrToDateFmt(const DateFormat, S: string): TDateTime;
4781: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
4782: //function DefDateFormat(AFourDigitYear: Boolean): string;
4783: //function DefDateMask(BlanksChar: Char; AFourDigitYear: Boolean): string;
4784: function FormatLongDate(Value: TDateTime): string;
4785: function FormatLongDateTime(Value: TDateTime): string;
4786: { end JvDateUtil }
4787: function BufToBinStr(Buf: Pointer; BufSize: Integer): string;
4788: function BinStrToBuf(Value: string; Buf: Pointer; BufSize: Integer): Integer;

```

```

4789: { begin JvStrUtils } { ** Common string handling routines ** }
4790: {$IFDEF UNIX}
4791: function iconversion(InP: PAnsiChar; OutP: Pointer; InBytes, OutBytes: Cardinal;
4792:   const ToCode, FromCode: AnsiString): Boolean;
4793: function iconvString(const S, ToCode, FromCode: AnsiString): string;
4794: function iconvWideString(const S: WideString; const ToCode, FromCode: AnsiString): WideString;
4795: function OemStrToAnsi(const S: AnsiString): AnsiString;
4796: function AnsiStrToOem(const S: AnsiString): AnsiString;
4797: {$ENDIF UNIX}
4798: function StrToOem(const AnsiStr: AnsiString): AnsiString;
4799: { StrToOem translates a string from the Windows character set into the OEM character set. }
4800: function OemToAnsiStr(const OemStr: AnsiString): AnsiString;
4801: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4802: function IsEmptyStr(const S: string; const EmptyChars: TSysCharSet): Boolean;
4803: { EmptyStr returns True if the given string contains only character from the EmptyChars. }
4804: function ReplaceStr(const S, Srch, Replace: string): string;
4805: { Returns string with every occurrence of Srch string replaced with Replace string. }
4806: function DelSpace(const S: string): string;
4807: { DelSpace return a string with all white spaces removed. }
4808: function DelChars(const S: string; Chr: Char): string;
4809: { DelChars return a string with all Chr characters removed. }
4810: function DelBSpace(const S: string): string;
4811: { DelBSpace trims leading spaces from the given string. }
4812: function DelEspace(const S: string): string;
4813: { DelEspace trims trailing spaces from the given string. }
4814: function DelRSpace(const S: string): string;
4815: { DelRSpace trims leading and trailing spaces from the given string. }
4816: function DelSpace1(const S: string): string;
4817: { DelSpace1 return a string with all non-single white spaces removed. }
4818: function Tab2Space(const S: string; Numb: Byte): string;
4819: { Tab2Space converts any tabulation character in the given string to the
4820:   Numb spaces characters. }
4821: function NPos(const C: string; S: string; N: Integer): Integer;
4822: { NPos searches for a N-th position of substring C in a given string. }
4823: function MakeStr(C: Char; N: Integer): string; overload;
4824: {$IFDEF COMPILER12_UP}
4825: function MakeStr(C: WideChar; N: Integer): WideString; overload;
4826: {$ENDIF !COMPILER12_UP}
4827: function MS(C: Char; N: Integer): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4828: { MakeStr return a string of length N filled with character C. }
4829: function AddChar(C: Char; const S: string; N: Integer): string;
4830: { AddChar return a string left-padded to length N with characters C. }
4831: function AddCharR(C: Char; const S: string; N: Integer): string;
4832: { AddCharR return a string right-padded to length N with characters C. }
4833: function LeftStr(const S: string; N: Integer): string;
4834: { LeftStr return a string right-padded to length N with blanks. }
4835: function RightStr(const S: string; N: Integer): string;
4836: { RightStr return a string left-padded to length N with blanks. }
4837: function CenterStr(const S: string; Len: Integer): string;
4838: { CenterStr centers the characters in the string based upon the Len specified. }
4839: function CompStr(const S1, S2: string): Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4840: { CompStr compares S1 to S2, with case-sensitivity. The return value is
4841:   -1 if S1 < S2, 0 if S1 = S2, or 1 if S1 > S2. }
4842: function CompText(const S1, S2: string): Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4843: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4844: function Copy2Symb(const S: string; Symb: Char): string;
4845: { Copy2Symb returns a substring of a string S from beginning to first character Symb. }
4846: function Copy2SymbDel(var S: string; Symb: Char): string;
4847: { Copy2SymbDel returns a substring of a string S from beginning to first
4848:   character Symb and removes this substring from S. }
4849: function Copy2Space(const S: string): string;
4850: { Copy2Symb returns a substring of a string S from beginning to first white space. }
4851: function Copy2SpaceDel(var S: string): string;
4852: { Copy2SpaceDel returns a substring of a string S from beginning to first
4853:   white space and removes this substring from S. }
4854: function AnsiProperCase(const S: string; const WordDelims: TSysCharSet): string;
4855: { Returns string, with the first letter of each word in uppercase,
4856:   all other letters in lowercase. Words are delimited by WordDelims. }
4857: function WordCount(const S: string; const WordDelims: TSysCharSet): Integer;
4858: { WordCount given a set of word delimiters, returns number of words in S. }
4859: function WordPosition(const N: Integer; const S: string; const WordDelims: TSysCharSet): Integer;
4860: { Given a set of word delimiters, returns start position of N'th word in S. }
4861: function ExtractWord(N: Integer; const S: string; const WordDelims: TSysCharSet): string;
4862: function ExtractWordPos(N: Integer; const S: string; const WordDelims: TSysCharSet; var Pos: Integer): string;
4863: function ExtractDelimited(N: Integer; const S: string; const Delims: TSysCharSet): string;
4864: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4865:   delimiters, return the N'th word in S. }
4866: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TSysCharSet): string;
4867: { ExtractSubstr given a set of word delimiters, returns the substring from S,
4868:   that started from position Pos. }
4869: function IsWordPresent(const W, S: string; const WordDelims: TSysCharSet): Boolean;
4870: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4871: function QuotedString(const S: string; Quote: Char): string;
4872: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4873: function ExtractQuotedString(const S: string; Quote: Char): string;
4874: { ExtractQuotedString removes the Quote characters from the beginning and
4875:   end of a quoted string, and reduces pairs of Quote characters within quoted string to single character. }
4876: function FindPart(const HelpWilds, InputStr: string): Integer;
4877: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }

```



```

4878: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4879: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4880: function XorString(const Key, Src: ShortString): ShortString;
4881: function XorEncode(const Key, Source: string): string;
4882: function XorDecode(const Key, Source: string): string;
4883: { ** Command line routines ** }
4884: function GetCmdLineArg(const Switch: string; ASwitchChars: TSysCharSet): string;
4885: { ** Numeric string handling routines ** }
4886: function Numb2USA(const S: string): string;
4887: { Numb2USA converts numeric string S to USA-format. }
4888: function Dec2Hex(N: Longint; A: Byte): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4889: { Dec2Hex converts the given value to a hexadecimal string representation
4890:   with the minimum number of digits (A) specified. }
4891: function Hex2Dec(const S: string): Longint;
4892: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4893: function Dec2Numb(N: Int64; A, B: Byte): string;
4894: { Dec2Numb converts the given value to a string representation with the
4895:   base equal to B and with the minimum number of digits (A) specified. }
4896: function Numb2Dec(S: string; B: Byte): Int64;
4897: { Numb2Dec converts the given B-based numeric string to the corresponding
4898:   integer value. }
4899: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4900: { IntToBin converts the given value to a binary string representation
4901:   with the minimum number of digits specified. }
4902: function IntToRoman(Value: Longint): string;
4903: { IntToRoman converts the given value to a roman numeric string representation. }
4904: function RomanToInt(const S: string): Longint;
4905: { RomanToInt converts the given string to an integer value. If the string
4906:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4907: function FindNotBlankCharPos(const S: string): Integer;
4908: function FindNotBlankCharPosW(const S: WideString): Integer;
4909: function AnsiChangeCase(const S: string): string;
4910: function WideChangeCase(const S: string): string;
4911: function StartsText(const SubStr, S: string): Boolean;
4912: function EndText(const SubStr, S: string): Boolean;
4913: function DequotedStr(const S: string; QuoteChar: Char = ''' ): string;
4914: function AnsiDequotedStr(const S: string; AQuote: Char): string; //follow Delphi 2009's "Ansi" prefix
4915: {end JvStrUtils}
4916: {$IFDEF UNIX}
4917: function GetTempFileName(const Prefix: AnsiString): AnsiString;
4918: {$ENDIF UNIX}
4919: { begin JvFileUtil }
4920: function FileDateTime(const FileName: string): TDateTime;
4921: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4922: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4923: function NormalDir(const DirName: string): string;
4924: function RemoveBackSlash(const DirName: string): string; // only for Windows/DOS Paths
4925: function ValidFileName(const FileName: string): Boolean;
4926: {$IFDEF MSWINDOWS}
4927: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4928: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4929: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4930: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4931: {$ENDIF MSWINDOWS}
4932: function GetWindowsDir: string;
4933: function GetSystemDir: string;
4934: function ShortToLongFileName(const ShortName: string): string;
4935: function LongToShortFileName(const LongName: string): string;
4936: function ShortToLongPath(const ShortName: string): string;
4937: function LongToShortPath(const LongName: string): string;
4938: {$IFDEF MSWINDOWS}
4939: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4940: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4941: {$ENDIF MSWINDOWS}
4942: { end JvFileUtil }
4943: // Works like PtInRect but includes all edges in comparison
4944: function PtInRectInclusive(R: TRect; Pt: TPoint): Boolean;
4945: // Works like PtInRect but excludes all edges from comparison
4946: function PtInRectExclusive(R: TRect; Pt: TPoint): Boolean;
4947: function FourDigitYear: Boolean; {$IFDEF SUPPORTS_DEPRECATED} deprecated; {$ENDIF}
4948: function IsFourDigitYear: Boolean;
4949: { moved from JvJCLUtils }
4950: //Open an object with the shell (url or something like that)
4951: function OpenObject(const Value: string): Boolean; overload;
4952: function OpenObject(Value: PChar): Boolean; overload;
4953: {$IFDEF MSWINDOWS}
4954: //Raise the last Exception
4955: procedure RaiseLastWin32; overload;
4956: procedure RaiseLastWin32(const Text: string); overload;
4957: //Raise the last Exception with a small comment from your part { GetFileVersion returns the most
4958:   significant 32 bits of a file's binary version number. Typically, this includes the major and minor
4959:   version placed together in one 32-bit Integer. I
4960: function GetFileVersion(const AFileName: string): Cardinal;
4961: {$EXTERNALSYM GetFileVersion}
4962: //Get version of Shell.dll
4963: function GetShellVersion: Cardinal;
4964: {$EXTERNALSYM GetShellVersion}
4965: // CD functions on HW
4966: procedure OpenCdDrive;

```

```

4965: procedure CloseCdDrive;
4966: // returns True if Drive is accessible
4967: function DiskInDrive(Drive: Char): Boolean;
4968: {$ENDIF MSWINDOWS}
4969: //Same as linux function ;)
4970: procedure PError(const Text: string);
4971: // execute a program without waiting
4972: procedure Exec(const FileName, Parameters, Directory: string);
4973: // execute a program and wait for it to finish
4974: function ExecuteAndWait(CmdLine: string; const WorkingDirectory: string; Visibility: Integer = SW_SHOW): Int;
4975: // returns True if this is the first instance of the program that is running
4976: function FirstInstance(const ATitle: string): Boolean;
4977: // restores a window based on it's classname and Caption. Either can be left empty
4978: // to widen the search
4979: procedure RestoreOtherInstance(const MainFormClassName, MainFormCaption: string);
4980: // manipulate the traybar and start button
4981: procedure HideTraybar;
4982: procedure ShowTraybar;
4983: procedure ShowStartButton(Visible: Boolean = True);
4984: // (rom) SC_MONITORPOWER is documented as Win 95 only(rom) better do some testing set monitor functions
4985: procedure MonitorOn;
4986: procedure MonitorOff;
4987: procedure LowPower;
4988: // send a key to the window named AppName
4989: function SendKey(const AppName: string; Key: Char): Boolean;
4990: {$IFDEF MSWINDOWS}
4991: // returns a list of all win currently visible, the Objects property is filled with their window handle
4992: procedure GetVisibleWindows(List: TStringList);
4993: // associates an extension to a specific program
4994: procedure AssociateExtension(const IconPath, ProgramName, Path, Extension: string);
4995: procedure AddToRecentDocs(const FileName: string);
4996: function GetRecentDocs: TStringList;
4997: {$ENDIF MSWINDOWS}
4998: function CharIsMoney(const Ch: Char): Boolean;
4999: //function StrToCurrDef(const Str: string; Def: Currency): Currency;
5000: function IntToExtended(I: Integer): Extended;
5001: { GetChangedText works out the new text given the current cursor pos & the key pressed
5002:   It is not very useful in other contexts, but in this unit as it is needed in both MemoEx and TypedEdit }
5003: function GetChangedText(const Text: string; SelStart, SelLength: Integer; Key: Char): string;
5004: function MakeYear4Digit(Year, Pivot: Integer): Integer;
5005: //function StrIsInteger(const S: string): Boolean;
5006: function StrIsFloatMoney(const Ps: string): Boolean;
5007: function StrIsDateTime(const Ps: string): Boolean;
5008: function PreformatDateString(Ps: string): string;
5009: function BooleanToInteger(const B: Boolean): Integer;
5010: function StringToBoolean(const Ps: string): Boolean;
5011: function SafeStrToDateTime(const Ps: string): TDateTime;
5012: function SafeStrToDate(const Ps: string): TDateTime;
5013: function SafeStrToTime(const Ps: string): TDateTime;
5014: function StrDelete(const psSub, psMain: string): string;
5015: { returns the fractional value of pcValue }
5016: function TimeOnly(pcValue: TDateTime): TTime;
5017: { returns the integral value of pcValue }
5018: function DateOnly(pcValue: TDateTime): TDate;
5019: type TdtKind = (dtkDateOnly, dtkTimeOnly, dtkDateTime);
5020: const { TDateTime value used to signify Null value }
5021:   NullEquivalentDate: TDateTime = 0.0;
5022: function DateIsNull(const pdtValue: TDateTime; const pdtKind: TdtKind): Boolean;
5023: // Replacement for Win32Check to avoid platform specific warnings in D6
5024: function OSCheck(RetVal: Boolean): Boolean;
5025: { Shortens a fully qualified Path name so that it can be drawn with a specified length limit.
5026:   Same as FileCtrl.MinimizeName in functionality (but not implementation). Included here to
5027:   not be forced to use FileCtrl unnecessarily }
5028: function MinimizeFileName(const FileName: string; Canvas: TCanvas; MaxLen: Integer): string;
5029: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
5030: { MinimizeString truncates long string, S, and appends'...'symbols,if Length of S is more than MaxLen }
5031: function MinimizeString(const S: string; const MaxLen: Integer): string;
5032: procedure RunDll32Internal(Wnd: THandle; const DLLName, FuncName, CmdLine: string; CmdShow: Integer =
  SW_SHOWDEFAULT);
5033: { GetDLLVersion loads DLLName, gets a pointer to DLLVersion function and calls it, returning major and
  minor version values from the function. Returns False if DLL not loaded or if GetDLLVersion couldn't be
  found. }
5034: function GetDLLVersion(const DLLName: string; var pdwMajor, pdwMinor: Integer): Boolean;
5035: {$ENDIF MSWINDOWS}
5036: procedure ResourceNotFound(ResID: PChar);
5037: function EmptyRect: TRect;
5038: function RectWidth(R: TRect): Integer;
5039: function RectHeight(R: TRect): Integer;
5040: function CompareRect(const R1, R2: TRect): Boolean;
5041: procedure RectNormalize(var R: TRect);
5042: function RectIsSquare(const R: TRect): Boolean;
5043: function RectSquare(var ARect: TRect; AMaxSize: Integer = -1): Boolean;
5044: //If AMaxSize = -1 ,then auto calc Square's max size
5045: {$IFDEF MSWINDOWS}
5046: procedure FreeUnusedOle;
5047: function GetWindowsVersion: string;
5048: function LoadDLL(const LibName: string): THandle;
5049: function RegisterServer(const ModuleName: string): Boolean;
5050: function UnregisterServer(const ModuleName: string): Boolean;

```

```

5051: { $ENDIF MSWINDOWS }
5052: { String routines }
5053: function GetEnvVar(const VarName: string): string;
5054: function AnsiUpperFirstChar(const S: string): string; //follow Delphi 2009's example with "Ansi" prefix
5055: function StringToPChar(var S: string): PChar;
5056: function StrPAlloc(const S: string): PChar;
5057: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
5058: function DropT(const S: string): string;
5059: { Memory routines }
5060: function AllocMemo(Size: Longint): Pointer;
5061: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
5062: procedure FreeMemo(var fpBlock: Pointer);
5063: function GetMemoSize(fpBlock: Pointer): Longint;
5064: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
5065: { Manipulate huge pointers routines }
5066: procedure HugeInc(var HugePtr: Pointer; Amount: Longint);
5067: procedure HugeDec(var HugePtr: Pointer; Amount: Longint);
5068: function HugeOffset(HugePtr: Pointer; Amount: Longint): Pointer;
5069: procedure HugeMove(Base: Pointer; Dst, Src, Size: Longint);
5070: procedure HMemCpy(DstPtr, SrcPtr: Pointer; Amount: Longint);
5071: function WindowClassName(Wnd: THandle): string;
5072: procedure SwitchToWindow(Wnd: THandle; Restore: Boolean);
5073: procedure ActivateWindow(Wnd: THandle);
5074: procedure ShowWinNoAnimate(Handle: THandle; CmdShow: Integer);
5075: procedure KillMessage(Wnd: THandle; Msg: Cardinal);
5076: { SetWindowTop put window to top without recreating window }
5077: procedure SetWindowTop(const Handle: THandle; const Top: Boolean);
5078: procedure CenterWindow(Wnd: THandle);
5079: function MakeVariant(const Values: array of Variant): Variant;
5080: { Convert dialog units to pixels and backwards }
5081: { $IFDEF MSWINDOWS }
5082: function DialogUnitsToPixelsX(DlgUnits: Word): Word;
5083: function DialogUnitsToPixelsY(DlgUnits: Word): Word;
5084: function PixelsToDialogUnitsX(PixUnits: Word): Word;
5085: function PixelsToDialogUnitsY(PixUnits: Word): Word;
5086: { $ENDIF MSWINDOWS }
5087: function GetUniqueFileNameInDir(const Path, FileNameMask: string): string;
5088: { $IFDEF BCB }
5089: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): THandle;
5090: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
5091: { $ELSE }
5092: function FindPrevInstance(const MainFormClass, ATitle: string): THandle;
5093: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
5094: { $ENDIF BCB }
5095: { $IFDEF MSWINDOWS }
5096: { BrowseForFolderNative displays Browse For Folder dialog }
5097: function BrowseForFolderNative(const Handle: THandle; const Title: string; var Folder: string): Boolean;
5098: { $ENDIF MSWINDOWS }
5099: procedure AntiAlias(Clip: TBitmap);
5100: procedure AntiAliasRect(Clip: TBitmap; XOrigin, YOrigin, XFinal, YFinal: Integer);
5101: procedure CopyRectDIBits(ACanvas: TCanvas; const DestRect: TRect;
5102:   ABitmap: TBitmap; const SourceRect: TRect);
5103: function IsTrueType(const FontName: string): Boolean;
5104: // Removes all non-numeric characters from AValue and returns the resulting string
5105: function TextToValText(const AValue: string): string;
5106: function ExecRegExpr(const ARegExpr, AInputStr: RegExprString): boolean;
5107: procedure SplitRegExpr(const ARegExpr, AInputStr: RegExprString; APieces: TStringList);
5108: function ReplaceRegExpr(const ARegExpr, AInputStr,
5109:   AReplaceStr: RegExprString; AUseSubstitution: boolean): RegExprString;
5109: function QuoteRegExprMetaChars(const AStr: RegExprString): RegExprString;
5110: function RegExprSubExpressions(const ARegExpr: string; ASubExprs: TStringList; AExtendedSyntax: boolean):
5111:   TStringList;
5112: *****unit uPSI_JvTFUtils;
5113: function JExtractYear( ADate: TDateTime): Word;
5114: function JExtractMonth( ADate: TDateTime): Word;
5115: function JExtractDay( ADate: TDateTime): Word;
5116: function JExtractHours( ATime: TDateTime): Word;
5117: function JExtractMins( ATime: TDateTime): Word;
5118: function JExtractSecs( ATime: TDateTime): Word;
5119: function JExtractMSecs( ATime: TDateTime): Word;
5120: function JFirstOfMonth( ADate: TDateTime): TDateTime;
5121: function JGetDayOfNthDOW( Year, Month, DOW, N: Word): Word;
5122: function JGetWeeksInMonth( Year, Month: Word; StartOfWeek: Integer): Word;
5123: procedure JIncBorlDOW( var BorlDOW: Integer; N: Integer);
5124: procedure JIncDOW( var DOW: TTFDayOfWeek; N: Integer);
5125: procedure JIncDays( var ADate: TDateTime; N: Integer);
5126: procedure JIncWeeks( var ADate: TDateTime; N: Integer);
5127: procedure JIncMonths( var ADate: TDateTime; N: Integer);
5128: procedure JIncYears( var ADate: TDateTime; N: Integer);
5129: function JEndOfMonth( ADate: TDateTime): TDateTime;
5130: function JIsFirstOfMonth( ADate: TDateTime): Boolean;
5131: function JIsEndOfMonth( ADate: TDateTime): Boolean;
5132: procedure JEnsureMonth( Month: Word);
5133: procedure JEnsureDOW( DOW: Word);
5134: function JEqualDates( D1, D2: TDateTime): Boolean;
5135: function JLesser( N1, N2: Integer): Integer;
5136: function JGreater( N1, N2: Integer): Integer;
5137: function JGetDivLength( TotalLength, DivCount, DivNum: Integer): Integer;
5138: function JGetDivNum( TotalLength, DivCount, X: Integer): Integer;

```

```

5139: Function GetDivStart( TotalLength, DivCount, DivNum : Integer) : Integer
5140: Function DOWToBorl( ADOW : TTFDayOfWeek) : Integer
5141: Function BorlToDOW( BorlDOW : Integer) : TTFDayOfWeek
5142: Function DateToDOW( ADate : TDateTime) : TTFDayOfWeek
5143: Procedure CalcTextPos( HostRect : TRect; var TextLeft, TextTop:Integer; var TextBounds : TRect;
AFont:TFont;AAngle: Integer; HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5144: Procedure DrawAngleText( ACanvas : TCanvas; HostRect : TRect; var TextBounds : TRect; AAngle : Integer;
HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5145: Function JRectWidth( ARect : TRect) : Integer
5146: Function JRectHeight( ARect : TRect) : Integer
5147: Function JEmptyRect : TRect
5148: Function IsClassByName( Obj : TObject; ClassName : string) : Boolean
5149:
5150: procedure SIRegister_MSUtils(CL: TPSPascalCompiler);
5151: begin
5152: Procedure HideTaskBarButton( hWindow : HWND)
5153: Function msLoadStr( ID : Integer) : String
5154: Function msFormat( fmt : String; params : array of const) : String
5155: Function msFileExists( const FileName : String) : Boolean
5156: Function msIntToStr( Int : Int64) : String
5157: Function msStrPas( const Str : PChar) : String
5158: Function msRenameFile( const OldName, NewName : String) : Boolean
5159: Function CutFileName( s : String) : String
5160: Function GetVersionInfo( var VersionString : String) : DWORD
5161: Function FormatTime( t : Cardinal) : String
5162: Function msCreateDir( const Dir : string) : Boolean
5163: Function SetAutoRun( NeedAutoRun : Boolean; AppName : String) : Boolean
5164: Function SetTreeViewStyle( const hTV : HWND; dwNewStyle : dword) : DWORD
5165: Function msStrLen( Str : PChar) : Integer
5166: Function msDirectoryExists( const Directory : String) : Boolean
5167: Function GetFolder( hWnd : hWnd; RootDir : Integer; Caption : String) : String
5168: Function SetBlendWindow( hWnd : HWND; AlphaBlend : Byte) : LongBool
5169: Function EditWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
5170: Procedure SetEditWndProc( hWnd : HWND; ptr : TObject)
5171: Function GetTextFromFile( Filename : String) : string
5172: Function IsTopMost( hWnd : HWND) : Bool // 'LWA_ALPHA','LongWord').SetUInt( $00000002);
5173: Function msStrToIntDef( const s : String; const i : Integer) : Integer
5174: Function msStrToInt( s : String) : Integer
5175: Function GetItemText( hDlg : THandle; ID : DWORD) : String
5176: end;
5177:
5178: procedure SIRegister_ESBMaths2(CL: TPSPascalCompiler);
5179: begin
5180: //TDynFloatArray', 'array of Extended
5181: TDynLWordArray', 'array of LongWord
5182: TDynLIntArray', 'array of LongInt
5183: TDynFloatMatrix', 'array of TDynFloatArray
5184: TDynLWordMatrix', 'array of TDynLWordArray
5185: TDynLIntMatrix', 'array of TDynLIntArray
5186: Function SquareAll( const X : TDynFloatArray) : TDynFloatArray
5187: Function InverseAll( const X : TDynFloatArray) : TDynFloatArray
5188: Function LnAll( const X : TDynFloatArray) : TDynFloatArray
5189: Function Log10All( const X : TDynFloatArray) : TDynFloatArray
5190: Function LinearTransform( const X : TDynFloatArray; Offset, Scale : Extended) : TDynFloatArray
5191: Function AddVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5192: Function SubVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5193: Function MultVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5194: Function DotProduct( const X, Y : TDynFloatArray) : Extended
5195: Function MNorm( const X : TDynFloatArray) : Extended
5196: Function MatrixIsRectangular( const X : TDynFloatMatrix) : Boolean
5197: Procedure MatrixDimensions(const X:TDynFloatMatrix;var Rows,Columns:LongWord;var Rectangular:Boolean;
5198: Function MatrixIsSquare( const X : TDynFloatMatrix) : Boolean
5199: Function MatricesSameDimensions( const X, Y : TDynFloatMatrix) : Boolean
5200: Function AddMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5201: Procedure AddToMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5202: Function SubtractMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5203: Procedure SubtractFromMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5204: Function MultiplyMatrixByConst( const X : TDynFloatMatrix; const K : Extended) : TDynFloatMatrix
5205: Procedure MultiplyMatrixByConst2( var X : TDynFloatMatrix; const K : Extended);
5206: Function MultiplyMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix;
5207: Function TransposeMatrix( const X : TDynFloatMatrix) : TDynFloatMatrix;
5208: Function GrandMean( const X : TDynFloatMatrix; var N : LongWord) : Extended
5209: end;
5210:
5211: procedure SIRegister_ESBMaths(CL: TPSPascalCompiler);
5212: begin
5213: 'ESBMinSingle','Single').setExtended( 1.5e-45);
5214: 'ESBMaxSingle','Single').setExtended( 3.4e+38);
5215: 'ESBMinDouble','Double').setExtended( 5.0e-324);
5216: 'ESBMaxDouble','Double').setExtended( 1.7e+308);
5217: 'ESBMinExtended','Extended').setExtended( 3.6e-4951);
5218: 'ESBMaxExtended','Extended').setExtended( 1.1e+4932);
5219: 'ESBMinCurrency','Currency').SetExtended( - 922337203685477.5807);
5220: 'ESBMaxCurrency','Currency').SetExtended( 922337203685477.5807);
5221: 'ESBSqrt2','Extended').setExtended( 1.4142135623730950488);
5222: 'ESBSqrt3','Extended').setExtended( 1.7320508075688772935);
5223: 'ESBSqrt5','Extended').setExtended( 2.2360679774997896964);
5224: 'ESBSqrt10','Extended').setExtended( 3.1622776601683793320);
5225: 'ESBSqrtPi','Extended').setExtended( 1.77245385090551602729);

```



```

5226: 'ESBCbrt2','Extended').setExtended( 1.2599210498948731648);
5227: 'ESBCbrt3','Extended').setExtended( 1.4422495703074083823);
5228: 'ESBCbrt10','Extended').setExtended( 2.1544346900318837219);
5229: 'ESBCbrt100','Extended').setExtended( 4.6415888336127788924);
5230: 'ESBCbrtPi','Extended').setExtended( 1.4645918875615232630);
5231: 'ESBInvSqrt2','Extended').setExtended( 0.70710678118654752440);
5232: 'ESBInvSqrt3','Extended').setExtended( 0.57735026918962576451);
5233: 'ESBInvSqrt5','Extended').setExtended( 0.44721359549995793928);
5234: 'ESBInvSqrtPi','Extended').setExtended( 0.56418958354775628695);
5235: 'ESBInvCbrtPi','Extended').setExtended( 0.68278406325529568147);
5236: 'ESBe','Extended').setExtended( 2.7182818284590452354);
5237: 'ESBe2','Extended').setExtended( 7.3890560989306502272);
5238: 'ESBePi','Extended').setExtended( 23.140692632779269006);
5239: 'ESBePiOn2','Extended').setExtended( 4.8104773809653516555);
5240: 'ESBePiOn4','Extended').setExtended( 2.1932800507380154566);
5241: 'ESBLn2','Extended').setExtended( 0.69314718055994530942);
5242: 'ESBLn10','Extended').setExtended( 2.30258509299404568402);
5243: 'ESBLnPi','Extended').setExtended( 1.14472988584940017414);
5244: 'ESBLog10Base2','Extended').setExtended( 3.3219280948873623478);
5245: 'ESBLog2Base10','Extended').setExtended( 0.30102999566398119521);
5246: 'ESBLog3Base10','Extended').setExtended( 0.47712125471966243730);
5247: 'ESBLogPiBase10','Extended').setExtended( 0.4971498726941339);
5248: 'ESBLogEBase10','Extended').setExtended( 0.43429448190325182765);
5249: 'ESBPi','Extended').setExtended( 3.1415926535897932385);
5250: 'ESBPiPi','Extended').setExtended( 3.1830988618379067154e-1);
5251: 'ESBTwoPi','Extended').setExtended( 6.2831853071795864769);
5252: 'ESBThreePi','Extended').setExtended( 9.4247779607693797153);
5253: 'ESBPi2','Extended').setExtended( 9.8696044010893586188);
5254: 'ESBPiToE','Extended').setExtended( 22.459157718361045473);
5255: 'ESBPiOn2','Extended').setExtended( 1.5707963267948966192);
5256: 'ESBPiOn3','Extended').setExtended( 1.0471975511965977462);
5257: 'ESBPiOn4','Extended').setExtended( 0.7853981633974483096);
5258: 'ESBThreePiOn2','Extended').setExtended( 4.7123889803846898577);
5259: 'ESBFourPiOn3','Extended').setExtended( 4.1887902047863909846);
5260: 'ESBTwoToPower63','Extended').setExtended( 9223372036854775808.0);
5261: 'ESBOneRadian','Extended').setExtended( 57.295779513082320877);
5262: 'ESBOneDegree','Extended').setExtended( 1.7453292519943295769E-2);
5263: 'ESBOneMinute','Extended').setExtended( 2.9088820866572159615E-4);
5264: 'ESBOneSecond','Extended').setExtended( 4.8481368110953599359E-6);
5265: 'ESBGamma','Extended').setExtended( 0.57721566490153286061);
5266: 'ESBLnRt2Pi','Extended').setExtended( 9.189385332046727E-1);
5267: //LongWord', 'Cardinal
5268: TBitList', 'Word
5269: Function UMul( const Num1, Num2 : LongWord) : LongWord
5270: Function UMulDiv2p32( const Num1, Num2 : LongWord) : LongWord
5271: Function UMulDiv( const Num1, Num2, Divisor : LongWord) : LongWord
5272: Function UMulMod( const Num1, Num2, Modulus : LongWord) : LongWord
5273: Function SameFloat( const X1, X2 : Extended) : Boolean
5274: Function FloatIsZero( const X : Extended) : Boolean
5275: Function FloatIsPositive( const X : Extended) : Boolean
5276: Function FloatIsNegative( const X : Extended) : Boolean
5277: Procedure IncLim( var B : Byte; const Limit : Byte)
5278: Procedure IncLimSI( var B : ShortInt; const Limit : ShortInt)
5279: Procedure IncLimW( var B : Word; const Limit : Word)
5280: Procedure IncLimI( var B : Integer; const Limit : Integer)
5281: Procedure IncLimL( var B : LongInt; const Limit : LongInt)
5282: Procedure DecLim( var B : Byte; const Limit : Byte)
5283: Procedure DecLimSI( var B : ShortInt; const Limit : ShortInt)
5284: Procedure DecLimW( var B : Word; const Limit : Word)
5285: Procedure DecLimI( var B : Integer; const Limit : Integer)
5286: Procedure DecLimL( var B : LongInt; const Limit : LongInt)
5287: Function MaxB( const B1, B2 : Byte) : Byte
5288: Function MinB( const B1, B2 : Byte) : Byte
5289: Function MaxSI( const B1, B2 : ShortInt) : ShortInt
5290: Function MinSI( const B1, B2 : ShortInt) : ShortInt
5291: Function MaxW( const B1, B2 : Word) : Word
5292: Function MinW( const B1, B2 : Word) : Word
5293: Function esbMaxI( const B1, B2 : Integer) : Integer
5294: Function esbMinI( const B1, B2 : Integer) : Integer
5295: Function MaxL( const B1, B2 : LongInt) : LongInt
5296: Function MinL( const B1, B2 : LongInt) : LongInt
5297: Procedure SwapB( var B1, B2 : Byte)
5298: Procedure SwapSI( var B1, B2 : ShortInt)
5299: Procedure SwapW( var B1, B2 : Word)
5300: Procedure SwapI( var B1, B2 : SmallInt)
5301: Procedure SwapL( var B1, B2 : LongInt)
5302: Procedure SwapI32( var B1, B2 : Integer)
5303: Procedure SwapC( var B1, B2 : LongWord)
5304: Procedure SwapInt64( var X, Y : Int64)
5305: Function esbSign( const B : LongInt) : ShortInt
5306: Function Max4Word( const X1, X2, X3, X4 : Word) : Word
5307: Function Min4Word( const X1, X2, X3, X4 : Word) : Word
5308: Function Max3Word( const X1, X2, X3 : Word) : Word
5309: Function Min3Word( const X1, X2, X3 : Word) : Word
5310: Function MaxBArray( const B : array of Byte) : Byte
5311: Function MaxWArray( const B : array of Word) : Word
5312: Function MaxSIArray( const B : array of ShortInt) : ShortInt
5313: Function MaxIArray( const B : array of Integer) : Integer
5314: Function MaxLArray( const B : array of LongInt) : LongInt

```

```

5315: Function MinBArray( const B : array of Byte ) : Byte
5316: Function MinWArray( const B : array of Word ) : Word
5317: Function MinSIArray( const B : array of ShortInt ) : ShortInt
5318: Function MiniArray( const B : array of Integer ) : Integer
5319: Function MinLArray( const B : array of LongInt ) : LongInt
5320: Function SumBArray( const B : array of Byte ) : Byte
5321: Function SumBArray2( const B : array of Byte ) : Word
5322: Function SumSIArray( const B : array of ShortInt ) : ShortInt
5323: Function SumSIArray2( const B : array of ShortInt ) : Integer
5324: Function SumWArray( const B : array of Word ) : Word
5325: Function SumWArray2( const B : array of Word ) : LongInt
5326: Function SumIArray( const B : array of Integer ) : Integer
5327: Function SumLArray( const B : array of LongInt ) : LongInt
5328: Function SumLWArray( const B : array of LongWord ) : LongWord
5329: Function ESBDigits( const X : LongWord ) : Byte
5330: Function BitsHighest( const X : LongWord ) : Integer
5331: Function ESBBitsNeeded( const X : LongWord ) : Integer
5332: Function esbGCD( const X, Y : LongWord ) : LongWord
5333: Function esbLCM( const X, Y : LongInt ) : Int64
5334: //Function esbLCM( const X, Y : LongInt ) : LongInt
5335: Function RelativePrime( const X, Y : LongWord ) : Boolean
5336: Function Get87ControlWord : TBitList
5337: Procedure Set87ControlWord( const CWord : TBitList )
5338: Procedure SwapExt( var X, Y : Extended )
5339: Procedure SwapDbl( var X, Y : Double )
5340: Procedure SwapSing( var X, Y : Single )
5341: Function esbSgn( const X : Extended ) : ShortInt
5342: Function Distance( const X1, Y1, X2, Y2 : Extended ) : Extended
5343: Function ExtMod( const X, Y : Extended ) : Extended
5344: Function ExtRem( const X, Y : Extended ) : Extended
5345: Function CompMOD( const X, Y : Comp ) : Comp
5346: Procedure Polar2XY( const Rho, Theta : Extended; var X, Y : Extended )
5347: Procedure XY2Polar( const X, Y : Extended; var Rho, Theta : Extended )
5348: Function DMS2Extended( const Degs, Mins, Secs : Extended ) : Extended
5349: Procedure Extended2DMS( const X : Extended; var Degs, Mins, Secs : Extended )
5350: Function MaxExt( const X, Y : Extended ) : Extended
5351: Function MinExt( const X, Y : Extended ) : Extended
5352: Function MaxEArray( const B : array of Extended ) : Extended
5353: Function MinEArray( const B : array of Extended ) : Extended
5354: Function MaxSArray( const B : array of Single ) : Single
5355: Function MinSArray( const B : array of Single ) : Single
5356: Function MaxCArray( const B : array of Comp ) : Comp
5357: Function MinCArray( const B : array of Comp ) : Comp
5358: Function SumSArray( const B : array of Single ) : Single
5359: Function SumEArray( const B : array of Extended ) : Extended
5360: Function SumSqEArray( const B : array of Extended ) : Extended
5361: Function SumSqDiffEArray( const B : array of Extended; Diff : Extended ) : Extended
5362: Function SumXYEArray( const X, Y : array of Extended ) : Extended
5363: Function SumCArray( const B : array of Comp ) : Comp
5364: Function FactorialX( A : LongWord ) : Extended
5365: Function PermutationX( N, R : LongWord ) : Extended
5366: Function esbBinomialCoeff( N, R : LongWord ) : Extended
5367: Function IsPositiveEArray( const X : array of Extended ) : Boolean
5368: Function esbGeometricMean( const X : array of Extended ) : Extended
5369: Function esbHarmonicMean( const X : array of Extended ) : Extended
5370: Function ESBMean( const X : array of Extended ) : Extended
5371: Function esbSampleVariance( const X : array of Extended ) : Extended
5372: Function esbPopulationVariance( const X : array of Extended ) : Extended
5373: Procedure esbSampleVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended )
5374: Procedure esbPopulationVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended )
5375: Function GetMedian( const SortedX : array of Extended ) : Extended
5376: Function GetMode( const SortedX : array of Extended; var Mode : Extended ) : Boolean
5377: Procedure GetQuartiles( const SortedX : array of Extended; var Q1, Q3 : Extended )
5378: Function ESBMagnitude( const X : Extended ) : Integer
5379: Function ESBTan( Angle : Extended ) : Extended
5380: Function ESBcot( Angle : Extended ) : Extended
5381: Function ESBcosec( const Angle : Extended ) : Extended
5382: Function ESBsec( const Angle : Extended ) : Extended
5383: Function ESBArcTan( X, Y : Extended ) : Extended
5384: Procedure ESBSinCos( Angle : Extended; var SinX, CosX : Extended )
5385: Function ESBArcCos( const X : Extended ) : Extended
5386: Function ESBArcSin( const X : Extended ) : Extended
5387: Function ESBArcSec( const X : Extended ) : Extended
5388: Function ESBArcCosec( const X : Extended ) : Extended
5389: Function ESBLog10( const X : Extended ) : Extended
5390: Function ESBLog2( const X : Extended ) : Extended
5391: Function ESBLogBase( const X, Base : Extended ) : Extended
5392: Function Pow2( const X : Extended ) : Extended
5393: Function IntPow( const Base : Extended; const Exponent : LongWord ) : Extended
5394: Function ESBIntPower( const X : Extended; const N : LongInt ) : Extended
5395: Function XtoY( const X, Y : Extended ) : Extended
5396: Function esbTenToY( const Y : Extended ) : Extended
5397: Function esbTwoToY( const Y : Extended ) : Extended
5398: Function LogXtoBaseY( const X, Y : Extended ) : Extended
5399: Function esbISqrt( const I : LongWord ) : LongWord
5400: Function ILog2( const I : LongWord ) : LongWord
5401: Function IGreatestPowerOf2( const N : LongWord ) : LongWord
5402: Function ESBArCosh( X : Extended ) : Extended
5403: Function ESBArSinh( X : Extended ) : Extended

```

```

5404: Function ESBarTanh( X : Extended) : Extended
5405: Function ESBcosh( X : Extended) : Extended
5406: Function ESBSinh( X : Extended) : Extended
5407: Function ESBTanh( X : Extended) : Extended
5408: Function InverseGamma( const X : Extended) : Extended
5409: Function esbGamma( const X : Extended) : Extended
5410: Function esbLnGamma( const X : Extended) : Extended
5411: Function esbBeta( const X, Y : Extended) : Extended
5412: Function IncompleteBeta( X : Extended; P, Q : Extended) : Extended
5413: end;
5414:
5415: *****Integer Huge Cardinal Utils*****
5416: Function Add_uint64_WithCarry( x, y : uint64; var Carry : Boolean) : uint64
5417: Function Add_uint32_WithCarry( x, y : uint32; var Carry : Boolean) : uint32
5418: Function Subtract_uint64_WithBorrow( x, y : uint64; var Borrow : Boolean) : uint64
5419: Function Subtract_uint32_WithBorrow( x, y : uint32; var Borrow : Boolean) : uint32
5420: Function BitCount_8( Value : byte) : integer
5421: Function BitCount_16( Value : uint16) : integer
5422: Function BitCount_32( Value : uint32) : integer
5423: Function BitCount_64( Value : uint64) : integer
5424: Function CountSetBits_64( Value : uint64) : integer TPrimalityTestNoticeProc',
5425: Procedure ( CountPrimalityTests : integer)
5426: Function gcd( a, b : THugeCardinal) : THugeCardinal
5427: Function lcm( a, b : THugeCardinal) : THugeCardinal
5428: Function isCoPrime( a, b : THugeCardinal) : boolean
5429: Function isProbablyPrime(p : THugeCardinal; OnProgress : TProgress; var wasAborted : boolean) : boolean
5430: Function hasSmallFactor( p : THugeCardinal) : boolean
5431: //Function GeneratePrime( NumBits : integer; OnProgress : TProgress; OnPrimalityTest:
TPrimalityTestNoticeProc; PassCount : integer; Pool1: TMemoryStreamPool; var Prime : THugeCardinal; var
NumbersTested : integer) : boolean
5432: Function Inverse( Prime, Modulus : THugeCardinal) : THugeCardinal
5433: Const ( 'StandardExponent', 'LongInt').SetInt( 65537);
5434: //Procedure Compute_RSA_Fundamentals_2Factors(RequiredBitLengthOfN:integer;Fixed_e:uint64;var N,e,d,
Totient:TProgress;OnPrimalityTest:TPrimalityTestNoticeProc;GeneratePrimePassCount:int;Pool1:TMemoryStreamPo
Numbers
5435: Function Validate_RSA_Fundamentals( var N, e, d, Totient : THugeCardinal) : boolean')
5436:
5437: procedure SIRegister_xrttl_math_Integer(CL: TPSPascalCompiler);
5438: begin
5439:   AddTypeS('TXRTLInteger', 'array of Integer
5440:   AddClassN(FindClass('TOBJECT'),'EXRTLMathException
5441:   (FindClass('TOBJECT'),'EXRTLExtendInvalidArgument
5442:   AddClassN(FindClass('TOBJECT'),'EXRTLDivisionByZero
5443:   AddClassN(FindClass('TOBJECT'),'EXRTLExpInvalidArgument
5444:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadix
5445:   AddClassN(FindClass('TOBJECT'),'EXRTLInvalidRadixDigit
5446:   AddClassN(FindClass('TOBJECT'),'EXRTLRootInvalidArgument
5447:   'BitsPerByte','LongInt').SetInt( 8);
5448:   BitsPerDigit,'LongInt').SetInt( 32);
5449:   SignBitMask,'LongWord').SetUInt( $80000000);
5450: Function XRTLAdjustBits( const ABits : Integer) : Integer
5451: Function XRTLLength( const AInteger : TXRTLInteger) : Integer
5452: Function XRTLDataBits( const AInteger : TXRTLInteger) : Integer
5453: Procedure XRTLBitPosition( const BitIndex : Integer; var Index, Mask : Integer)
5454: Procedure XRTLBitSet( var AInteger : TXRTLInteger; const BitIndex : Integer)
5455: Procedure XRTLBitReset( var AInteger : TXRTLInteger; const BitIndex : Integer)
5456: Function XRTLBitGet( const AInteger : TXRTLInteger; const BitIndex : Integer) : Integer
5457: Function XRTLBitGetBool( const AInteger : TXRTLInteger; const BitIndex : Integer) : Boolean
5458: Function XRTLExtend(const AInteger:TXRTLInteger;ADataBits:Integer;Sign:Int;var AResult:TXRTLInteger):Int;
5459: Function XRTLZeroExtend(const AInteger:TXRTLInteger;ADataBits:Integer; var AResult:TXRTLInteger):Integer;
5460: Function XRTLSignExtend(const AInteger:TXRTLInteger; ADataBits:Integer;var AResult:TXRTLInteger):Integer;
5461: Function XRTLSignStrip(const AInteger:TXRTLInteger;var AResult:TXRTLInteger;const AMinDataBits:Int):Int;
5462: Procedure XRTLNot( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5463: Procedure XRTLOr( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5464: Procedure XRTLEnd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5465: Procedure XRTLXor( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5466: Function XRTLSign( const AInteger : TXRTLInteger) : Integer
5467: Procedure XRTLZero( var AInteger : TXRTLInteger)
5468: Procedure XRTLOne( var AInteger : TXRTLInteger)
5469: Procedure XRTLMOne( var AInteger : TXRTLInteger)
5470: Procedure XRTLTwo( var AInteger : TXRTLInteger)
5471: Function XRTLNeg( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5472: Function XRTLAbs( const AInteger : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5473: Procedure XRTLFullSum( const A, B, C : Integer; var Sum, Carry : Integer)
5474: Function XRTLAdd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5475: Function XRTLAdd1( const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5476: Function XRTLSub( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer;
5477: Function XRTLSub1( const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5478: Function XRTLCompare( const AInteger1, AInteger2 : TXRTLInteger) : Integer;
5479: Function XRTLCompare1( const AInteger1 : TXRTLInteger; const AInteger2 : Int64) : Integer;
5480: Function XRTLUmul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5481: Function XRTLMulAdd(const AInteger1,AInteger2,AInteger3:TXRTLInteger; var AResult:TXRTLInteger):Integer
5482: Function XRTLMul( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger) : Integer
5483: Procedure XRTLDivMod( const AInteger1, AInteger2 : TXRTLInteger; var QResult, RResult : TXRTLInteger)
5484: Procedure XRTLSqr( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5485: Procedure XRTLSqrt( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5486: Procedure XRTLRoot( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5487: Procedure XRTLRootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
AHighApproxResult:TXRTLInteger)

```

```

5488: Procedure XRTLURootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
      AHighApproxResult:TXRTLInteger);
5489: Procedure XRTLExp( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5490: Procedure XRTLExpMod( const AInteger1, AInteger2, AInteger3 : TXRTLInteger;var AResult: TXRTLInteger)
5491: Procedure XRTLSLBB(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5492: Procedure XRTLSABL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5493: Procedure XRTLRCBL(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5494: Procedure XRTLSLDL(const AInteger:TXRTLInteger;const DigitCount:Integer; var AResult:TXRTLInteger)
5495: Procedure XRTLSADL(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5496: Procedure XRTLRCDL(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5497: Procedure XRTLSLBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5498: Procedure XRTLSABR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5499: Procedure XRTLRCBR(const AInteger: TXRTLInteger; const BitCount:Integer; var AResult: TXRTLInteger)
5500: Procedure XRTLSLDR(const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult: TXRTLInteger)
5501: Procedure XRTLSADR(const AInteger: TXRTLInteger; const DigitCount:Integer;var AResult: TXRTLInteger)
5502: Procedure XRTLRCDR(const AInteger: TXRTLInteger;const DigitCount:Integer;var AResult: TXRTLInteger)
5503: Function XRTLToHex( const AInteger : TXRTLInteger; Digits : Integer) : string
5504: Function XRTLToBin( const AInteger : TXRTLInteger; Digits : Integer) : string
5505: Function XRTLToString( const AInteger : TXRTLInteger; Radix : Integer; Digits : Integer) : string
5506: Procedure XRTLFromHex( const Value : string; var AResult : TXRTLInteger)
5507: Procedure XRTLFromBin( const Value : string; var AResult : TXRTLInteger)
5508: Procedure XRTLFromString( const Value : string; var AResult : TXRTLInteger; Radix : Integer)
5509: Procedure XRTLAssign( const AInteger : TXRTLInteger; var AResult : TXRTLInteger);
5510: Procedure XRTLAssign1( const Value : Integer; var AResult : TXRTLInteger);
5511: Procedure XRTLAssign2( const Value : Int64; var AResult : TXRTLInteger);
5512: Procedure XRTLAppend( const ALow, AHigh : TXRTLInteger; var AResult : TXRTLInteger)
5513: Procedure XRTLSplit(const AInteger: TXRTLInteger; var ALow,AHigh: TXRTLInteger;LowDigits: Integer)
5514: Function XRTLGetMSBitIndex( const AInteger : TXRTLInteger) : Integer
5515: Procedure XRTLMinMax(const AInteger1, AInteger2 : TXRTLInteger;var AMinResult,AMaxResult: TXRTLInteger)
5516: Procedure XRTLMin( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5517: Procedure XRTLMin1( const AInteger1: TXRTLInteger;const AInteger2:Integer;var AResult: TXRTLInteger);
5518: Procedure XRTLMax( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5519: Procedure XRTLMax1(const AInteger1:TXRTLInteger; const AInteger2:Integer; var AResult:TXRTLInteger);
5520: Procedure XRTLGCD( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5521: Procedure XRTLSwap( var AInteger1, AInteger2 : TXRTLInteger)
5522: Procedure XRTLFactorial( const AInteger: TXRTLInteger; var AResult : TXRTLInteger)
5523: Procedure XRTLFactorialMod( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5524: end;
5525:
5526: procedure SIRegister_JvXPCoreUtils(CL: TPSPascalCompiler);
5527: begin
5528: Function JvXPMethodsEqual( const Method1, Method2 : TMethod) : Boolean
5529: Procedure JvXPDrawLine( const ACanvas : TCanvas; const X1, Y1, X2, Y2 : Integer)
5530: Procedure JvXPCreateGradientRect( const AWidth, AHeight : Integer; const StartColor, EndColor : TColor;
      const Colors:TJvXPGradientColors;const Style:TJvXPGradientStyle;const Dithered:Boolean;var Bitmap:TBitmap);
5531: Procedure JvXPAdjustBoundRect(const BorderWidth:Byte; const ShowBoundLines:Boolean; const
      BoundLines:TJvXPBoundLines; var Rect : TRect)
5532: Procedure JvXPDrawBoundLines(const ACanvas:TCanvas;const BoundLines:TJvXPBoundLines;const
      AColor:TColor;const Rect:TRect);
5533: Procedure JvXPConvertToGray2( Bitmap : TBitmap)
5534: Procedure JvXPRenderText( const AParent : TControl; const ACanvas : TCanvas; ACaption : TCaption; const
      AFont : TFont; const AEnabled, AShowAccelChar : Boolean; var ARect : TRect; AFlags : Integer)
5535: Procedure JvXPFrame3D(const ACanvas:TCanvas;const Rect:TRect;const TopColor,BottomColor:TColor;const
      Swapped:Boolean);
5536: Procedure JvXPColorizeBitmap( Bitmap : TBitmap; const AColor : TColor)
5537: Procedure JvXPSetDrawFlags(const AAlignment: TAlignment;const AWordWrap: Boolean; var Flags : Integer)
5538: Procedure JvXPPlaceText( const AParent: TControl; const ACanvas : TCanvas; const AText : TCaption; const
      AFont : TFont; const AEnabled,AShowAccelChar:Boolean;const AAlignment:TAlignment;const
      AWordWrap:Boolean;var Rect:TRect)
5539: end;
5540:
5541:
5542: procedure SIRegister_uwinstr(CL: TPSPascalCompiler);
5543: begin
5544: Function StrDec( S : String) : String
5545: Function uIsNumeric( var S : String; var X : Float) : Boolean
5546: Function ReadNumFromEdit( Edit : TEdit) : Float
5547: Procedure WriteNumToFile( var F : Text; X : Float)
5548: end;
5549:
5550: procedure SIRegister_utexplot(CL: TPSPascalCompiler);
5551: begin
5552: Function TeX_InitGraphics( FileName : String; PgWidth, PgHeight : Integer; Header : Boolean) : Boolean
5553: Procedure TeX_SetWindow( X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
5554: Procedure TeX_LeaveGraphics( Footer : Boolean)
5555: Procedure TeX_SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
5556: Procedure TeX_SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
5557: Procedure TeX_SetGraphTitle( Title : String)
5558: Procedure TeX_SetOxTitle( Title : String)
5559: Procedure TeX_SetOyTitle( Title : String)
5560: Procedure TeX_PlotOxAxis
5561: Procedure TeX_PlotOyAxis
5562: Procedure TeX_PlotGrid( Grid : TGrid)
5563: Procedure TeX_WriteGraphTitle
5564: Function TeX_SetMaxCurv( NCurv : Byte) : Boolean
5565: Procedure TeX_SetPointParam( CurvIndex, Symbol, Size : Integer)
5566: Procedure TeX_SetLineParam( CurvIndex, Style : Integer; Width : Float; Smooth : Boolean)
5567: Procedure TeX_SetCurvLegend( CurvIndex : Integer; Legend : String)
5568: Procedure TeX_SetCurvStep( CurvIndex, Step : Integer)

```



```

5569: Procedure TeX_PlotCurve( X, Y : TVector; Lb, Ub, CurvIndex : Integer)
5570: Procedure TeX_PlotCurveWithErrorBars( X, Y, S : TVector; Ns, Lb, Ub, CurvIndex : Integer)
5571: Procedure TeX_PlotFunc( Func : TFunc; X1, X2 : Float; Npt : Integer; CurvIndex : Integer)
5572: Procedure TeX_WriteLegend( NCurv : Integer; ShowPoints, ShowLines : Boolean)
5573: Procedure TeX_ConRec( Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
5574: Function Xcm( X : Float ) : Float
5575: Function Ycm( Y : Float ) : Float
5576: end;
5577:
5578: *-----*)
5579: procedure SIRegister_VarRecUtils(CL: TPSPascalCompiler);
5580: begin
5581:   TConstArray', 'array of TVarRec
5582:   Function CopyVarRec( const Item : TVarRec ) : TVarRec
5583:   Function CreateConstArray( const Elements : array of const ) : TConstArray
5584:   Procedure FinalizeVarRec( var Item : TVarRec)
5585:   Procedure FinalizeConstArray( var Arr : TConstArray)
5586: end;
5587:
5588: procedure SIRegister_StStrS(CL: TPSPascalCompiler);
5589: begin
5590:   Function HexBS( B : Byte ) : ShortString
5591:   Function HexWS( W : Word ) : ShortString
5592:   Function HexLS( L : LongInt ) : ShortString
5593:   Function HexPtrS( P : Pointer ) : ShortString
5594:   Function BinaryBS( B : Byte ) : ShortString
5595:   Function BinaryWS( W : Word ) : ShortString
5596:   Function BinaryLS( L : LongInt ) : ShortString
5597:   Function OctalBS( B : Byte ) : ShortString
5598:   Function OctalWS( W : Word ) : ShortString
5599:   Function OctalLS( L : LongInt ) : ShortString
5600:   Function Str2Int16S( const S : ShortString; var I : SmallInt ) : Boolean
5601:   Function Str2WordS( const S : ShortString; var I : Word ) : Boolean
5602:   Function Str2LongS( const S : ShortString; var I : LongInt ) : Boolean
5603:   Function Str2RealS( const S : ShortString; var R : Double ) : Boolean
5604:   Function Str2Reals( const S : ShortString; var R : Real ) : Boolean
5605:   Function Str2ExtS( const S : ShortString; var R : Extended ) : Boolean
5606:   Function Long2StrS( L : LongInt ) : ShortString
5607:   Function Real2StrS( R : Double; Width : Byte; Places : ShortInt ) : ShortString
5608:   Function Ext2StrS( R : Extended; Width : Byte; Places : ShortInt ) : ShortString
5609:   Function ValPrepS( const S : ShortString ) : ShortString
5610:   Function CharStrS( C : AnsiChar; Len : Cardinal ) : ShortString
5611:   Function PadChS( const S : ShortString; C : AnsiChar; Len : Cardinal ) : ShortString
5612:   Function PadS( const S : ShortString; Len : Cardinal ) : ShortString
5613:   Function LeftPadChS( const S : ShortString; C : AnsiChar; Len : Cardinal ) : ShortString
5614:   Function LeftPadS( const S : ShortString; Len : Cardinal ) : ShortString
5615:   Function TrimLeadS( const S : ShortString ) : ShortString
5616:   Function TrimTrails( const S : ShortString ) : ShortString
5617:   Function TrimS( const S : ShortString ) : ShortString
5618:   Function TrimSpacesS( const S : ShortString ) : ShortString
5619:   Function CenterChS( const S : ShortString; C : AnsiChar; Len : Cardinal ) : ShortString
5620:   Function CenterS( const S : ShortString; Len : Cardinal ) : ShortString
5621:   Function EntabS( const S : ShortString; TabSize : Byte ) : ShortString
5622:   Function DetabS( const S : ShortString; TabSize : Byte ) : ShortString
5623:   Function ScrambleS( const S, Key : ShortString ) : ShortString
5624:   Function SubstituteS( const S, FromStr, ToStr : ShortString ) : ShortString
5625:   Function FilterS( const S, Filters : ShortString ) : ShortString
5626:   Function CharExistsS( const S : ShortString; C : AnsiChar ) : Boolean
5627:   Function CharCountS( const S : ShortString; C : AnsiChar ) : Byte
5628:   Function WordCountS( const S, WordDelims : ShortString ) : Cardinal
5629:   Function WordPositionS( N : Cardinal; const S, WordDelims : ShortString; var Pos : Cardinal ) : Boolean
5630:   Function ExtractWords( N : Cardinal; const S, WordDelims : ShortString ) : ShortString
5631:   Function AsciiCountS( const S, WordDelims : ShortString; Quote : AnsiChar ) : Cardinal
5632:   Function AsciiPositionS( N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar;var Pos:Cardinal):Boolean
5633:   Function ExtractAsciiS( N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar): ShortString
5634:   Procedure WordWrapS(const InSt: ShortString; var OutSt,Overlap: ShortString;
Margin:Cardinal;PadToMargin:Boolean)
5635:   Function CompStringS( const S1, S2 : ShortString ) : Integer
5636:   Function CompUCStringS( const S1, S2 : ShortString ) : Integer
5637:   Function SoundexS( const S : ShortString ) : ShortString
5638:   Function MakeLetterSetS( const S : ShortString ) : Longint
5639:   Procedure BMMakeTableS( const MatchString : ShortString; var BT : BTable)
5640:   Function BMSearchS(var Buffer,BufLength:Card;var BT:BTable;const MatchString:ShortString;var
Pos:Cardinal):Bool;
5641:   Function BMSearchUCS(var Buffer,BufLength:Card;var BT:BTable;const MatchString:ShortString;var
Pos:Cardinal):Bool;
5642:   Function DefaultExtensionS( const Name, Ext : ShortString ) : ShortString
5643:   Function ForceExtensionS( const Name, Ext : ShortString ) : ShortString
5644:   Function JustFileNameS( const PathName : ShortString ) : ShortString
5645:   Function JustNameS( const PathName : ShortString ) : ShortString
5646:   Function JustExtensionS( const Name : ShortString ) : ShortString
5647:   Function JustPathNameS( const PathName : ShortString ) : ShortString
5648:   Function AddBackSlashS( const DirName : ShortString ) : ShortString
5649:   Function CleanPathNameS( const PathName : ShortString ) : ShortString
5650:   Function HasExtensionS( const Name : ShortString; var DotPos : Cardinal ) : Boolean
5651:   Function CommaizeS( L : LongInt ) : ShortString
5652:   Function CommaizeChS( L : Longint; Ch : AnsiChar ) : ShortString
5653:   Function FloatFormS(const Mask:ShortString;R:TstFloat;const LtCurr,RtCurr:ShortString;Sep,
DecPt:Char):ShortString;

```

```

5654: Function LongIntFormS(const Mask:ShortString;L:LongInt;const LtCurr,
RtCurr:ShortString;Sep:AnsiChar):ShortString;
5655: Function StrChPosS( const P : ShortString; C : AnsiChar; var Pos : Cardinal) : Boolean
5656: Function StrStPosS( const P, S : ShortString; var Pos : Cardinal) : Boolean
5657: Function StrStCopyS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5658: Function StrChInsertS( const S : ShortString; C : AnsiChar; Pos : Cardinal) : ShortString
5659: Function StrStInsertS( const S1, S2 : ShortString; Pos : Cardinal) : ShortString
5660: Function StrChDeleteS( const S : ShortString; Pos : Cardinal) : ShortString
5661: Function StrStDeleteS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5662: Function ContainsOnlyS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5663: Function ContainsOtherThanS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5664: Function CopyLeftS( const S : ShortString; Len : Cardinal) : ShortString
5665: Function CopyMidS( const S : ShortString; First, Len : Cardinal) : ShortString
5666: Function CopyRightS( const S : ShortString; First : Cardinal) : ShortString
5667: Function CopyRightAbsS( const S : ShortString; NumChars : Cardinal) : ShortString
5668: Function CopyFromNthWordS(const S,WordDelims:string;const AWord:String;N:Card;var
SubString:ShortString):Bool;
5669: Function DeleteFromNthWordS(const S,WordDelims:String;AWord:ShortString;N:Card;var
SubStr:ShortString):Bool;
5670: Function CopyFromToWordS(const S,WordDelims,Word1,Word2:ShortString;N1,N2:Card;var
SubString:ShortString):Bool;
5671: Function DeleteFromToWordS(const S,WordDelims,Wrd1,Wrd2:ShortString;N1,N2:Card;var
SubString:ShortString):Bool;
5672: Function CopyWithinS( const S, Delimiter : ShortString; Strip : Boolean) : ShortString
5673: Function DeleteWithinS( const S, Delimiter : ShortString) : ShortString
5674: Function ExtractTokensS(const S,
Delims:ShortString;QuoteChar:AnsiChar;AllowNulls:Boolean;Tokens:TStrings):Cardinal
5675: Function IsChAlphaS( C : Char) : Boolean
5676: Function IsChNumericS( C : Char; const Numbers : ShortString) : Boolean
5677: Function IsChAlphaNumericS( C : Char; const Numbers : ShortString) : Boolean
5678: Function IsStrAlphaS( const S : ShortString) : Boolean
5679: Function IsStrNumericS( const S, Numbers : ShortString) : Boolean
5680: Function IsStrAlphaNumericS( const S, Numbers : ShortString) : Boolean
5681: Function LastWords( const S, WordDelims, AWord : ShortString; var Position : Cardinal) : Boolean
5682: Function LastWordAbsS( const S, WordDelims : ShortString; var Position : Cardinal) : Boolean
5683: Function LastStringS( const S, AString : ShortString; var Position : Cardinal) : Boolean
5684: Function LeftTrimCharsS( const S, Chars : ShortString) : ShortString
5685: Function KeepCharsS( const S, Chars : ShortString) : ShortString
5686: Function RepeatStringS(const RepeatString:ShortString;var Repetitions: Cardinal; MaxLen :
Cardinal):ShortString;
5687: Function ReplaceStringS(const S,OldString,NewString:ShortString;N:Cardinal;var
Replacements:Cardinal):ShortString;
5688: Function ReplaceStringAllS(const S,OldString,NewString:ShortString;var Replacements:Cardinal):ShortString;
5689: Function ReplaceWordsS(const S,WordDelims,OldWord,NewW:SString;N:Cardinal;var
Replacements:Cardinal):ShortString
5690: Function ReplaceWordAllS(const S,WordDelims,OldWord,NewWord:ShortString;var
Replacements:Cardinal):ShortString
5691: Function RightTrimCharsS( const S, Chars : ShortString) : ShortString
5692: Function StrWithinS(const S,SearchStr: ShortString;Start:Cardinal;var Position:Cardinal):boolean
5693: Function TrimCharsS( const S, Chars : ShortString) : ShortString
5694: Function WordPosS(const S,WordDelims,AWord:ShortString;N:Cardinal; var Position: Cardinal):Boolean
5695: end;
5696:
5697:
5698: *****unit uPSI_StUtils; from Systools4*****
5699: Function SignL( L : LongInt) : Integer
5700: Function SignF( F : Extended) : Integer
5701: Function MinWord( A, B : Word) : Word
5702: Function MidWord( W1, W2, W3 : Word) : Word
5703: Function MaxWord( A, B : Word) : Word
5704: Function MinLong( A, B : LongInt) : LongInt
5705: Function MidLong( L1, L2, L3 : LongInt) : LongInt
5706: Function MaxLong( A, B : LongInt) : LongInt
5707: Function MinFloat( F1, F2 : Extended) : Extended
5708: Function MidFloat( F1, F2, F3 : Extended) : Extended
5709: Function MaxFloat( F1, F2 : Extended) : Extended
5710: Function MakeInteger16( H, L : Byte) : SmallInt
5711: Function MakeWordS( H, L : Byte) : Word
5712: Function SwapNibble( B : Byte) : Byte
5713: Function SwapWord( L : LongInt) : LongInt
5714: Procedure SetFlag( var Flags : Word; FlagMask : Word)
5715: Procedure ClearFlag( var Flags : Word; FlagMask : Word)
5716: Function FlagIsSet( Flags, FlagMask : Word) : Boolean
5717: Procedure SetByteFlag( var Flags : Byte; FlagMask : Byte)
5718: Procedure ClearByteFlag( var Flags : Byte; FlagMask : Byte)
5719: Function ByteFlagIsSet( Flags, FlagMask : Byte) : Boolean
5720: Procedure SetLongFlag( var Flags : LongInt; FlagMask : LongInt)
5721: Procedure ClearLongFlag( var Flags : LongInt; FlagMask : LongInt)
5722: Function LongFlagIsSet( Flags, FlagMask : LongInt) : Boolean
5723: Procedure ExchangeBytes( var I, J : Byte)
5724: Procedure ExchangeWords( var I, J : Word)
5725: Procedure ExchangeLongInts( var I, J : LongInt)
5726: Procedure ExchangeStructs( var I, J, Size : Cardinal)
5727: Procedure FillWord( var Dest, Count : Cardinal; Filler : Word)
5728: Procedure FillStruct( var Dest, Count : Cardinal; var Filler, FillerSize : Cardinal)
5729: Function AddWordToPtr( P : __Pointer; W : Word) : __Pointer
5730: //*****uPSI_StFIN;*****
5731: Function AccruedInterestMaturity(Issue,Maturity:TStDate;Rate,Par:Extended;Basis: TStBasis): Extended
5732: Function AccruedInterestPeriodic(Issue,Settlement,Maturity:TStDate;Rate,
Par:Extended;Frequency:TStFrequency; Basis : TStBasis) : Extended

```

```

5733: Function BondDuration( Settlement,Maturity:TStDate;Rate,
Yield:Ext;Frequency:TStFrequency;Basis:TStBasis):Extended;
5734: Function BondPrice(Settlement,Maturity:TStDate;Rate,Yield,
Redemption:Ext;Freq:TStFrequency;Basis:TStBasis): Extended
5735: Function CumulativeInterest( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
Integer; Frequency : TStFrequency; Timing : TStPaymentTime ) : Extended
5736: Function CumulativePrincipal( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
Integer; Frequency : TStFrequency; Timing : TStPaymentTime ) : Extended
5737: Function DayCount( Day1, Day2 : TStDate; Basis : TStBasis ) : LongInt
5738: Function DecliningBalance( Cost, Salvage : Extended; Life, Period, Month : Integer ) : Extended
5739: Function DiscountRate(Settlement,Maturity:TStDate; Price,Redemption:Extended;Basis:TStBasis): Extended;
5740: Function DollarToDecimal( FracDollar : Extended; Fraction : Integer ) : Extended
5741: Function DollarToDecimalText( DecDollar : Extended ) : string
5742: Function DollarToFraction( DecDollar : Extended; Fraction : Integer ) : Extended
5743: Function DollarToFractionStr( FracDollar : Extended; Fraction : Integer ) : string
5744: Function EffectiveInterestRate( NominalRate : Extended; Frequency : TStFrequency ) : Extended
5745: Function FutureValueS(Rate:Extended;NPeriods:Int;Pmt,
PV:Extended;Freq:TStFrequency;Timing:TStPaymentTime):Extended;
5746: Function FutureValueSchedule( Principal : Extended; const Schedule : array of Double ) : Extended
5747: Function FutureValueSchedule16( Principal : Extended; const Schedule, NRates : Integer ) : Extended
5748: Function InterestRates(NPeriods:Int;Pmt,PV,
FV:Extended;Freq:TStFrequency;Timing:TStPaymentTime;Guess:Extended):Extend;
5749: Function InternalRateOfReturn( const Values : array of Double; Guess : Extended ) : Extended
5750: Function InternalRateOfReturn16( const Values, NValues : Integer; Guess : Extended ) : Extended
5751: Function IsCardValid( const S : string ) : Boolean
5752: Function ModifiedDuration(Settlement,Maturity:TStDate;Rate,
Yield:Extended;Freq:TStFrequency;Basis:TStBasis):Extended;
5753: Function ModifiedIRR(const Values : array of Double; FinanceRate,ReinvestRate: Extended ) : Extended
5754: Function ModifiedIRR16(const Values, NValues:Integer; FinanceRate,ReinvestRate: Extended ) : Extended
5755: Function NetPresentValueS( Rate : Extended; const Values : array of Double ) : Extended
5756: Function NetPresentValue16( Rate : Extended; const Values, NValues : Integer ) : Extended
5757: Function NominalInterestRate( EffectRate : Extended; Frequency : TStFrequency ) : Extended
5758: Function NonperiodicIRR(const Values:array of Double;const Dates:array of TStDate;Guess:Extended):Extended;
5759: Function NonperiodicNPV(Rate:Extended;const Values:array of Double;const Dates:array of TStDate):Extended;
5760: Function Payment( Rate : Extended; NPeriods : Integer; PV, FV : Extended; Frequency : TStFrequency; Timing
: TStPaymentTime): Extended
5761: Function Periods(Rate:Extended;Pmt,PV,FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Integer;
5762: Function PresentValueS( Rate: Extended; NPeriods: Integer; Pmt, FV: Extended; Frequency: TStFrequency;
Timing: TStPaymentTime): Extended
5763: Function ReceivedAtMaturity(Settlement,Maturity:TStDate;Invest,Discount:Extended;Basis:TStBasis):Extended;
5764: Function RoundToDecimal( Value : Extended; Places : Integer; Bankers : Boolean ) : Extended
5765: Function TBillEquivYield( Settlement, Maturity : TStDate; Discount : Extended ) : Extended
5766: Function TBillPrice( Settlement, Maturity : TStDate; Discount : Extended ) : Extended
5767: Function TBillYield( Settlement, Maturity : TStDate; Price : Extended ) : Extended
5768: Function VariableDecliningBalance( Cost, Salvage : Extended; Life : Integer; StartPeriod, EndPeriod,
Factor : Extended; NoSwitch : boolean ) : Extended
5769: Function YieldDiscounted(Settlement,Maturity:TStDate;Price,Redemption:Extended;Basis:TStBasis):Extended;
5770: Function YieldPeriodic(Settlement,Maturity:TStDate;Rate,Price,
Redemption:Extended;Freq:TStFrequency;Basis:TStBasis): Extended
5771: Function YieldMaturity(Issue,Settlement,Maturity:TStDate;Rate,Price:Extended;Basis:TStBasis):Extended;
5772:
5773: *****unit uPSI_StAstroP;
5774: Procedure PlanetsPos( JD : Double; var PA : TStPlanetsArray)
5775: *****unit unit uPSI_StStat; Statistic Package of SysTools*****
5776: Function AveDev( const Data : array of Double ) : Double
5777: Function AveDev16( const Data, NData : Integer ) : Double
5778: Function Confidence( Alpha, StandardDev : Double; Size : LongInt ) : Double
5779: Function Correlation( const Data1, Data2 : array of Double ) : Double
5780: Function Correlation16( const Data1, Data2, NData : Integer ) : Double
5781: Function Covariance( const Data1, Data2 : array of Double ) : Double
5782: Function Covariance16( const Data1, Data2, NData : Integer ) : Double
5783: Function DevSq( const Data : array of Double ) : Double
5784: Function DevSql6( const Data, NData : Integer ) : Double
5785: Procedure Frequency(const Data:array of Double;const Bins:array of Double;var Counts: array of LongInt);
5786: ///Procedure Frequency16( const Data, NData : Integer; const Bins, NBins : Integer; var Counts)
5787: Function GeometricMeanS( const Data : array of Double ) : Double
5788: Function GeometricMean16( const Data, NData : Integer ) : Double
5789: Function HarmonicMeanS( const Data : array of Double ) : Double
5790: Function HarmonicMean16( const Data, NData : Integer ) : Double
5791: Function Largest( const Data : array of Double; K : Integer ) : Double
5792: Function Largest16( const Data, NData : Integer; K : Integer ) : Double
5793: Function MedianS( const Data : array of Double ) : Double
5794: Function Median16( const Data, NData : Integer ) : Double
5795: Function Mode( const Data : array of Double ) : Double
5796: Function Model16( const Data, NData : Integer ) : Double
5797: Function Percentile( const Data : array of Double; K : Double ) : Double
5798: Function Percentile16( const Data, NData : Integer; K : Double ) : Double
5799: Function PercentRank( const Data : array of Double; X : Double ) : Double
5800: Function PercentRank16( const Data, NData : Integer; X : Double ) : Double
5801: Function Permutations( Number, NumberChosen : Integer ) : Extended
5802: Function Combinations( Number, NumberChosen : Integer ) : Extended
5803: Function Factorials( N : Integer ) : Extended
5804: Function Rank( Number : Double; const Data : array of Double; Ascending : Boolean ) : Integer
5805: Function Rank16( Number : Double; const Data, NData : Integer; Ascending : Boolean ) : Integer
5806: Function Smallest( const Data : array of Double; K : Integer ) : Double
5807: Function Smallest16( const Data, NData : Integer; K : Integer ) : Double
5808: Function TrimMean( const Data : array of Double; Percent : Double ) : Double
5809: Function TrimMean16( const Data, NData : Integer; Percent : Double ) : Double
5810: AddTypeS('TStLinEst', 'record B0 : Double; B1 : double; seB0 : double; seB'

```

```

5811:   +1 : Double; R2 : Double; sigma :Double; SSR: double; SSE: Double; F0 : Double; df : Integer;end
5812: Procedure LinEst(const KnownY:array of Double;const KnownX:array of Double;var
    LF:TStLinEst;ErrorStats:Bool;
5813: Procedure LogEst(const KnownY:array of Double;const KnownX:array of Double;var
    LF:TStLinEst;ErrorStats:Bool;
5814: Function Forecast(X: Double; const KnownY: array of Double; const KnownX: array of Double) : Double
5815: Function ForecastExponential(X:Double;const KnownY:array of Double;const KnownX:array of Double):Double;
5816: Function Intercept( const KnownY : array of Double; const KnownX : array of Double) : Double
5817: Function RSquared( const KnownY : array of Double; const KnownX : array of Double) : Double
5818: Function Slope( const KnownY : array of Double; const KnownX : array of Double) : Double
5819: Function StandardErrorY( const KnownY : array of Double; const KnownX : array of Double) : Double
5820: Function BetaDist( X, Alpha, Beta, A, B : Single) : Single
5821: Function BetaInv( Probability, Alpha, Beta, A, B : Single) : Single
5822: Function BinomDist( NumberS, Trials : Integer; ProbabilityS : Single; Cumulative : Boolean) : Single
5823: Function CritBinom( Trials : Integer; ProbabilityS, Alpha : Single) : Integer
5824: Function ChiDist( X : Single; DegreesFreedom : Integer) : Single
5825: Function ChiInv( Probability : Single; DegreesFreedom : Integer) : Single
5826: Function ExponDist( X, Lambda : Single; Cumulative : Boolean) : Single
5827: Function FDist( X : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5828: Function FInv( Probability : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5829: Function LogNormDist( X, Mean, StandardDev : Single) : Single
5830: Function LogInv( Probability, Mean, StandardDev : Single) : Single
5831: Function NormDist( X, Mean, StandardDev : Single; Cumulative : Boolean) : Single
5832: Function NormInv( Probability, Mean, StandardDev : Single) : Single
5833: Function NormSDist( Z : Single) : Single
5834: Function NormSInv( Probability : Single) : Single
5835: Function Poisson( X : Integer; Mean : Single; Cumulative : Boolean) : Single
5836: Function TDist( X : Single; DegreesFreedom : Integer; TwoTails : Boolean) : Single
5837: Function TInv( Probability : Single; DegreesFreedom : Integer) : Single
5838: Function Erfc( X : Single) : Single
5839: Function GammaLn( X : Single) : Single
5840: Function LargestSort( const Data : array of Double; K : Integer) : Double
5841: Function SmallestSort( const Data : array of double; K : Integer) : Double
5842:
5843: procedure SIRegister_TStSorter(CL: TPSPascalCompiler);
5844:   Function OptimumHeapToUse( RecLen : Cardinal; NumRecs : LongInt) : LongInt
5845:   Function MinimumHeapToUse( RecLen : Cardinal) : LongInt
5846:   Function MergeInfo( MaxHeap : LongInt; RecLen : Cardinal; NumRecs : LongInt) : TMergeInfo
5847:   Function DefaultMergeName( MergeNum : Integer) : string
5848:   Procedure ArraySort( var A, RecLen, NumRecs : Cardinal; Compare : TUntypedCompareFunc)
5849:
5850: procedure SIRegister_StAstro(CL: TPSPascalCompiler);
5851: Function AmountOfSunlight( LD : TStDate; Longitude, Latitude : Double) : TStTime
5852: Function FixedRiseSet( LD : TStDate; RA, DC, Longitude, Latitude : Double) : TStRiseSetRec
5853: Function SunPos( UT : TStDateTimeRec) : TStPosRec
5854: Function SunPosPrim( UT : TStDateTimeRec) : TStSunXYZRec
5855: Function SunRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5856: Function Twilight( LD : TStDate; Longitude, Latitude : Double; TwiType:TStTwilight):TStRiseSetRec
5857: Function LunarPhase( UT : TStDateTimeRec) : Double
5858: Function MoonPos( UT : TStDateTimeRec) : TStMoonPosRec
5859: Function MoonRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5860: Function FirstQuarter( D : TStDate) : TStLunarRecord
5861: Function FullMoon( D : TStDate) : TStLunarRecord
5862: Function LastQuarter( D : TStDate) : TStLunarRecord
5863: Function NewMoon( D : TStDate) : TStLunarRecord
5864: Function NextFirstQuarter( D : TStDate) : TStDateTimeRec
5865: Function NextFullMoon( D : TStDate) : TStDateTimeRec
5866: Function NextLastQuarter( D : TStDate) : TStDateTimeRec
5867: Function NextNewMoon( D : TStDate) : TStDateTimeRec
5868: Function PrevFirstQuarter( D : TStDate) : TStDateTimeRec
5869: Function PrevFullMoon( D : TStDate) : TStDateTimeRec
5870: Function PrevLastQuarter( D : TStDate) : TStDateTimeRec
5871: Function PrevNewMoon( D : TStDate) : TStDateTimeRec
5872: Function SiderealTime( UT : TStDateTimeRec) : Double
5873: Function Solstice( Y, Epoch : Integer; Summer : Boolean) : TStDateTimeRec
5874: Function Equinox( Y, Epoch : Integer; Vernal : Boolean) : TStDateTimeRec
5875: Function SEaster( Y, Epoch : Integer) : TStDate
5876: Function DateTimeToAJD( D : TDateTime) : Double
5877: Function HoursMin( RA : Double) : ShortString
5878: Function DegsMin( DC : Double) : ShortString
5879: Function AJDToDateTime( D : Double) : TDateTime
5880:
5881: Procedure SIRegister_StDate(CL: TPSPascalCompiler);
5882: Function CurrentDate : TStDate
5883: Function StValidDate( Day, Month, Year, Epoch : Integer) : Boolean
5884: Function DMYtoStDate( Day, Month, Year, Epoch : Integer) : TStDate
5885: Procedure StDateToDMY( Julian : TStDate; var Day, Month, Year : Integer)
5886: Function StIncDate( Julian : TStDate; Days, Months, Years : Integer) : TStDate
5887: Function IncDateTrunc( Julian : TStDate; Months, Years : Integer) : TStDate
5888: Procedure StDateDiff( Date1, Date2 : TStDate; var Days, Months, Years : Integer)
5889: Function BondDateDiff( Date1, Date2 : TStDate; DayBasis : TStBondDateType) : TStDate
5890: Function WeekOfYear( Julian : TStDate) : Byte
5891: Function AstJulianDate( Julian : TStDate) : Double
5892: Function AstJulianDatetoStDate( AstJulian : Double; Truncate : Boolean) : TStDate
5893: Function AstJulianDatePrim( Year, Month, Date : Integer; UT : TStTime) : Double
5894: Function StDayOfWeek( Julian : TStDate) : TStDayType
5895: Function DayOfWeekDMY( Day, Month, Year, Epoch : Integer) : TStDayType
5896: Function StIsLeapYear( Year : Integer) : Boolean
5897: Function StDaysInMonth( Month : Integer; Year, Epoch : Integer) : Integer

```



```

5898: Function ResolveEpoch( Year, Epoch : Integer) : Integer
5899: Function ValidTime( Hours, Minutes, Seconds : Integer) : Boolean
5900: Procedure StTimeToHMS( T : TStTime; var Hours, Minutes, Seconds : Byte)
5901: Function HMStoStTime( Hours, Minutes, Seconds : Byte) : TStTime
5902: Function CurrentTime : TStTime
5903: Procedure TimeDiff( Time1, Time2 : TStTime; var Hours, Minutes, Seconds : Byte)
5904: Function StIncTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5905: Function DecTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5906: Function RoundToNearestHour( T : TStTime; Truncate : Boolean) : TStTime
5907: Function RoundToNearestMinute( const T : TStTime; Truncate : Boolean) : TStTime
5908: Procedure DateTimeDiff(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;var Days:LongInt;var Secs:LongInt
5909: Procedure IncDateTime(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;Days:Integer;Secs:LongInt)
5910: Function DateTimeToStDate( DT : TDateTime) : TStDate
5911: Function DateTimeToStTime( DT : TDateTime) : TStTime
5912: Function StDateToDateTime( D : TStDate) : TDateTime
5913: Function StTimeToDateTime( T : TStTime) : TDateTime
5914: Function Convert2ByteDate( TwoByteDate : Word) : TStDate
5915: Function Convert4ByteDate( FourByteDate : TStDate) : Word
5916:
5917: Procedure SIRegister_StDateSt(CL: TPSPascalCompiler);
5918: Function DateStringHMStoAstJD( const Picture, DS : string; H, M, S, Epoch : integer) : Double
5919: Function MonthToString( const Month : Integer) : string
5920: Function DateStringToStDate( const Picture, S : string; Epoch : Integer) : TStDate
5921: Function DateStringToDMY(const Picture,S:string; Epoch:Integer; var D, M, Y : Integer):Boolean
5922: Function StDateToDateString( const Picture : string; const Julian : TStDate; Pack : Boolean):string
5923: Function DayOfWeekToString( const WeekDay : TStDayType) : string
5924: Function DMYtoDateString(const Picture:string;Day,Month,Year,Epoch:Integer;Pack:Boolean): string);
5925: Function CurrentDateString( const Picture : string; Pack : Boolean) : string
5926: Function CurrentTimeString( const Picture : string; Pack : Boolean) : string
5927: Function TimeStringToHMS( const Picture, St : string; var H, M, S : Integer) : Boolean
5928: Function TimeStringToStTime( const Picture, S : string) : TStTime
5929: Function StTimeToAmPmString( const Picture : string; const T : TStTime; Pack : Boolean) : string
5930: Function StTimeToTimeString( const Picture : string; const T : TStTime; Pack : Boolean) : string
5931: Function DateStringIsBlank( const Picture, S : string) : Boolean
5932: Function InternationalDate( ForceCentury : Boolean) : string
5933: Function InternationalLongDate( ShortNames : Boolean; ExcludeDOW : Boolean) : string
5934: Function InternationalTime( ShowSeconds : Boolean) : string
5935: Procedure ResetInternationalInfo
5936:
5937: procedure SIRegister_StBase(CL: TPSPascalCompiler);
5938: Function DestroyNode( Container : TStContainer; Node : TStNode; OtherData : Pointer) : Boolean
5939: Function AnsiUpperCaseShort32( const S : string) : string
5940: Function AnsiCompareTextShort32( const S1, S2 : string) : Integer
5941: Function AnsiCompareStrShort32( const S1, S2 : string) : Integer
5942: Function HugeCompressRLE( const InBuffer, InLen : Longint; var OutBuffer) : Longint
5943: Function HugeDecompressRLE( const InBuffer, InLen : Longint; var OutBuffer,OutLen:LongInt): Longint
5944: Procedure HugeFillChar( var Dest, Count : Longint; Value : Byte)
5945: Procedure HugeFillStruc( var Dest, Count : Longint; const Value, ValSize : Cardinal)
5946: Function Upcase( C : AnsiChar) : AnsiChar
5947: Function LoCase( C : AnsiChar) : AnsiChar
5948: Function CompareLetterSets( Set1, Set2 : LongInt) : Cardinal
5949: Function CompStruct( const S1, S2, Size : Cardinal) : Integer
5950: Function Search(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardinal):Bool;
5951: Function StSearch(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
5952: Function SearchUC(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
5953: Function IsOrInheritsFrom( Root, Candidate : TClass) : boolean
5954: Procedure RaiseContainerError( Code : longint)
5955: Procedure RaiseContainerErrorFmt( Code : Longint; Data : array of const)
5956: Function ProductOverflow( A, B : LongInt) : Boolean
5957: Function StNewStr( S : string) : PShortString
5958: Procedure StDisposeStr( PS : PShortString)
5959: Procedure ValLongInt( S : ShortString; var LI : Longint; var ErrorCode : integer)
5960: Procedure ValSmallint( const S : ShortString; var SI : smallint; var ErrorCode : integer)
5961: Procedure ValWord( const S : ShortString; var Wd : word; var ErrorCode : integer)
5962: Procedure RaiseStError( ExceptionClass : EStExceptionClass; Code : LongInt)
5963: Procedure RaiseStWin32Error( ExceptionClass : EStExceptionClass; Code : LongInt)
5964: Procedure RaiseStWin32ErrorEx( ExceptionClass : EStExceptionClass; Code : LongInt; Info : string)
5965:
5966: procedure SIRegister_usvd(CL: TPSPascalCompiler);
5967: begin
5968: Procedure SV_DeComp( A : TMatrix; Lb, Ub1, Ub2 : Integer; S : TVector; V : TMatrix)
5969: Procedure SV_SetZero( S : TVector; Lb, Ub : Integer; Tol : Float)
5970: Procedure SV_Solve(U:TMatrix; S:TVector;V:TMatrix;B:TVector;Lb,Ub1,Ub2:Integer;X:TVector);
5971: Procedure SV_Approx( U : TMatrix; S : TVector; V : TMatrix; Lb, Ub1, Ub2 : Integer; A : TMatrix)
5972: Procedure RKF45(F:TDiffEqs;Neqn:Int;Y,Yp:TVector;var T:Float;Tout,RelErr,AbsErr:Float;var Flag:Int;
5973: end;
5974:
5975: *****unit unit ; StMath Package of SysTools*****
5976: Function IntPowerS( Base : Extended; Exponent : Integer) : Extended
5977: Function PowerS( Base, Exponent : Extended) : Extended
5978: Function StInvCos( X : Double) : Double
5979: Function StInvSin( Y : Double) : Double
5980: Function StInvTan2( X, Y : Double) : Double
5981: Function StTan( A : Double) : Double
5982: Procedure DumpException; unit StExpEng;
5983: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
5984:
5985: *****unit unit ; StCRC Package of SysTools*****
5986: Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt

```

```

5987: Function Adler32OfStream( Stream : TStream; CurCrc : LongInt ) : LongInt
5988: Function Adler32OfFile( FileName : AnsiString ) : LongInt
5989: Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal ) : Cardinal
5990: Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal ) : Cardinal
5991: Function Crc16OfFile( FileName : AnsiString ) : Cardinal
5992: Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt ) : LongInt
5993: Function Crc32OfStream( Stream : TStream; CurCrc : LongInt ) : LongInt
5994: Function Crc32OfFile( FileName : AnsiString ) : LongInt
5995: Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal ) : Cardinal
5996: Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal ) : Cardinal
5997: Function InternetSumOfFile( FileName : AnsiString ) : Cardinal
5998: Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal ) : Cardinal
5999: Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal ) : Cardinal
6000: Function Kermit16OfFile( FileName : AnsiString ) : Cardinal
6001:
6002: //*****unit unit ; StBCD Package of SysTools*****
6003: Function AddBcd( const B1, B2 : TbcdS ) : TbcdS
6004: Function SubBcd( const B1, B2 : TbcdS ) : TbcdS
6005: Function MulBcd( const B1, B2 : TbcdS ) : TbcdS
6006: Function DivBcd( const B1, B2 : TbcdS ) : TbcdS
6007: Function ModBcd( const B1, B2 : TbcdS ) : TbcdS
6008: Function NegBcd( const B : TbcdS ) : TbcdS
6009: Function AbsBcd( const B : TbcdS ) : TbcdS
6010: Function FracBcd( const B : TbcdS ) : TbcdS
6011: Function IntBcd( const B : TbcdS ) : TbcdS
6012: Function RoundDigitsBcd( const B : TbcdS; Digits : Cardinal ) : TbcdS
6013: Function RoundPlacesBcd( const B : TbcdS; Places : Cardinal ) : TbcdS
6014: Function ValBcd( const S : string ) : TbcdS
6015: Function LongBcd( L : LongInt ) : TbcdS
6016: Function ExtBcd( E : Extended ) : TbcdS
6017: Function ExpBcd( const B : TbcdS ) : TbcdS
6018: Function LnBcd( const B : TbcdS ) : TbcdS
6019: Function IntPowBcd( const B : TbcdS; E : LongInt ) : TbcdS
6020: Function PowBcd( const B, E : TbcdS ) : TbcdS
6021: Function SqrtBcd( const B : TbcdS ) : TbcdS
6022: Function CmpBcd( const B1, B2 : TbcdS ) : Integer
6023: Function EqDigitsBcd( const B1, B2 : TbcdS; Digits : Cardinal ) : Boolean
6024: Function EqPlacesBcd( const B1, B2 : TbcdS; Digits : Cardinal ) : Boolean
6025: Function IsIntBcd( const B : TbcdS ) : Boolean
6026: Function TruncBcd( const B : TbcdS ) : LongInt
6027: Function BcdExt( const B : TbcdS ) : Extended
6028: Function RoundBcd( const B : TbcdS ) : LongInt
6029: Function StrBcd( const B : TbcdS; Width, Places : Cardinal ) : string
6030: Function StrExpBcd( const B : TbcdS; Width : Cardinal ) : string
6031: Function FormatBcd( Format : string; const B : TbcdS ) : string
6032: Function StrGeneralBcd( const B : TbcdS ) : string
6033: Function FloatFormBcd( const Mask: string; B: TbcdS; const LtCurr, RtCurr: string; Sep, DecPt: AnsiChar ) : string
6034: Procedure ConvertBcd( const SrcB, SrcSize : Byte; var DestB, DestSize : Byte)
6035:
6036: //*****unit unit ; StTxtDat; TStTextDataRecordSet Package of SysTools*****
6037: Procedure StParseLine( const Data : AnsiString; Schema : TStTextDataSchema; Result : TStrings)
6038: Function StFieldTypeToStr( FieldType : TStSchemaFieldType ) : AnsiString
6039: Function StStrToFieldType( const S : AnsiString ) : TStSchemaFieldType
6040: Function StDeEscape( const EscStr : AnsiString ) : Char
6041: Function StDoEscape( Delim : Char ) : AnsiString
6042: Function StTrimTrailingChars( const S : AnsiString; Trailer : Char ) : AnsiString
6043: Function AnsiHashText( const S : string; Size : Integer ) : Integer
6044: Function AnsiHashStr( const S : string; Size : Integer ) : Integer
6045: Function AnsiELFHashText( const S : string; Size : Integer ) : Integer
6046: Function AnsiELFHashStr( const S : string; Size : Integer ) : Integer
6047:
6048: //*****unit unit ; StNetCon Package of SysTools*****
6049: with AddClassN( FindClass( 'TStComponent' ), 'TStNetConnection' ) do begin
6050:   Constructor Create( AOwner : TComponent )
6051:   Function Connect : DWord
6052:   Function Disconnect : DWord
6053:   RegisterProperty( 'Password', 'String', iptrw );
6054:   Property( 'UserName', 'String', iptrw );
6055:   Property( 'ConnectOptions', 'TStNetConnectOptionsSet', iptrw );
6056:   Property( 'DisconnectOptions', 'TStNetDisconnectOptionsSet', iptrw );
6057:   Property( 'LocalDevice', 'String', iptrw );
6058:   Property( 'ServerName', 'String', iptrw );
6059:   Property( 'ShareName', 'String', iptrw );
6060:   Property( 'OnConnect', 'TNotifyEvent', iptrw );
6061:   Property( 'OnConnectFail', 'TOnConnectFailEvent', iptrw );
6062:   Property( 'OnConnectCancel', 'TOnConnectCancelEvent', iptrw );
6063:   Property( 'OnDisconnect', 'TNotifyEvent', iptrw );
6064:   Property( 'OnDisconnectFail', 'TOnDisconnectFailEvent', iptrw );
6065:   Property( 'OnDisconnectCancel', 'TOnDisconnectCancelEvent', iptrw );
6066: end;
6067: //*****Thread Functions Context of Win API --- more objects in SyncObjs.pas
6068: / /153 unit uPSI_SyncObjs, unit uPSIParallelJobs;
6069: Procedure InitializeCriticalSection( var lpCriticalSection : TRTLCriticalSection )
6070: Procedure EnterCriticalSection( var lpCriticalSection : TRTLCriticalSection )
6071: Procedure LeaveCriticalSection( var lpCriticalSection : TRTLCriticalSection )
6072: Function InitializeCriticalSectionAndSpinCount( var
  lpCriticalSection: TRTLCriticalSection; dwSpinCount: DWORD ): BOOL;
6073: Function SetCriticalSectionSpinCount( var lpCriticalSection: TRTLCriticalSection; dwSpinCount: DWORD ): DWORD;
6074: Function TryEnterCriticalSection( var lpCriticalSection : TRTLCriticalSection ) : BOOL

```

```

6075: Procedure DeleteCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6076: Function GetThreadContext( hThread : THandle; var lpContext : TContext) : BOOL
6077: Function SetThreadContext( hThread : THandle; const lpContext : TContext) : BOOL
6078: Function SuspendThread( hThread : THandle) : DWORD
6079: Function ResumeThread( hThread : THandle) : DWORD
6080: Function CreateThread2(ThreadFunc: TThreadFunction2; thrId: DWord) : THandle
6081: Function GetCurrentThread : THandle
6082: Procedure ExitThread( dwExitCode : DWORD)
6083: Function TerminateThread( hThread : THandle; dwExitCode : DWORD) : BOOL
6084: Function GetExitCodeThread( hThread : THandle; var lpExitCode : DWORD) : BOOL
6085: Procedure EndThread(ExitCode: Integer);
6086: Function WaitForSingleObject( hHandle : THandle; dwMilliseconds : DWORD) : DWORD
6087: Function MakeProcInstance( Proc : FARPROC; Instance : THandle) : FARPROC
6088: Procedure FreeProcInstance( Proc : FARPROC)
6089: Procedure FreeLibraryAndExitThread( hLibModule : HMODULE; dwExitCode : DWORD)
6090: Function DisableThreadLibraryCalls( hLibModule : HMODULE) : BOOL
6091: Procedure ParallelJob( ASelf : TObject; ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6092: Procedure ParallelJob1( ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6093: Procedure ParallelJob2(AJobGroup:TJobsGroup;ASelf:TObject;ATarget:Ptr;AParam:Pointer;ASafeSection:bool;
6094: Procedure ParallelJob3( AJobGroup:TJobsGroup;ATarget:Pointer;AParam:Pointer;ASafeSection: boolean);
6095: Function CreateParallelJob(ASelf:TObject;ATarget:Pointer;AParam:Ptr;ASafeSection:bool:TParallelJob;
6096: Function CreateParallelJob1(ATarget:Pointer; AParam:Pointer; ASafeSection : boolean) : TParallelJob;
6097: Function CurrentParallelJobInfo : TParallelJobInfo
6098: Function ObtainParallelJobInfo : TParallelJobInfo
6099:
6100: *****unit uPSI_JclMime;
6101: Function MimeEncodeString( const S : AnsiString) : AnsiString
6102: Function MimeDecodeString( const S : AnsiString) : AnsiString
6103: Procedure MimeEncodeStream( const InputStream : TStream; const OutputStream : TStream)
6104: Procedure MimeDecodeStream( const InputStream : TStream; const OutputStream : TStream)
6105: Function MimeEncodedSize( const I : Cardinal) : Cardinal
6106: Function MimeDecodedSize( const I : Cardinal) : Cardinal
6107: Procedure MimeEncode( var InputBuffer : string; const InputByteCount : Cardinal; var OutputBuffer)
6108: Function MimeDecode(var InputBuffer:string;const InputBytesCount:Cardinal;var OutputBuffer):Cardinal;
6109: Function MimeDecodePartial(var InputBuffer:string;const InputBytesCount:Cardinal;var
OutputBuffer:string;var ByteBuffer : Cardinal; var ByteBufferSize : Cardinal) : Cardinal
6110: Function MimeDecodePartialEnd(var OutputBuf:string;const ByteBuf:Card;const ByteBufferSize:Card):
Cardinal;
6111:
6112: *****unit uPSI_JclPrint;
6113: Procedure DirectPrint( const Printer, Data : string)
6114: Procedure SetPrinterPixelsPerInch
6115: Function GetPrinterResolution : TPoint
6116: Function CharFitsWithinDots( const Text : string; const Dots : Integer) : Integer
6117: Procedure PrintMemo( const Memo : TMemo; const Rect : TRect)
6118:
6119:
6120: //*****unit uPSI_ShLwApi;*****
6121: Function StrChr( lpStart : PChar; wMatch : WORD) : PChar
6122: Function StrChrI( lpStart : PChar; wMatch : WORD) : PChar
6123: Function StrCmpN( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6124: Function StrCmpNI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6125: Function StrCSpn( lpStr_, lpSet : PChar) : Integer
6126: Function StrCSpnI( lpStr1, lpSet : PChar) : Integer
6127: Function StrDup( lpSrch : PChar) : PChar
6128: Function StrFormatByteSize( dw : DWORD; szBuf : PChar; uiBufSize : UINT) : PChar
6129: Function StrFormatKBSIZE( qdw : Dword; szBuf : PChar; uiBufSize : UINT) : PChar
6130: Function StrFromTimeInterval(pszOut: PChar; cchMax:UINT;dwTimeMS:DWORD; digits: Integer): Integer
6131: Function StrIsIntlEqual( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6132: Function StrNCat( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6133: Function StrPBrk( psz, pszSet : PChar) : PChar
6134: Function StrRChr( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6135: Function StrRChrI( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6136: Function StrRStrI( lpSource, lpLast, lpSrch : PChar) : PChar
6137: Function StrSpn( psz, pszSet : PChar) : Integer
6138: Function StrStr( lpFirst, lpSrch : PChar) : PChar
6139: Function StrStrI( lpFirst, lpSrch : PChar) : PChar
6140: Function StrToInt( lpSrch : PChar) : Integer
6141: Function StrToIntEx( pszString : PChar; dwFlags : DWORD; var piRet : Integer) : BOOL
6142: Function StrTrim( psz : PChar; pszTrimChars : PChar) : BOOL
6143: Function ChrCmpI( w1, w2 : WORD) : BOOL
6144: Function ChrCmpIA( w1, w2 : WORD) : BOOL
6145: Function ChrCmpIW( w1, w2 : WORD) : BOOL
6146: Function StrIntlEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6147: Function StrIntlEqNI( s1, s2 : PChar; nChar : Integer) : BOOL
6148: Function StrCatBuff( pszDest, pszSrc : PChar; cchDestBuffSize : Integer) : PChar
6149: Function StrCpyNX( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6150: Function IntlStrEqWorker( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6151: Function IntlStrEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6152: SZ_CONTENTTYPE_HTMLA('String').SetString( 'text/html
6153: SZ_CONTENTTYPE_HTMLW('String').SetString( 'text/html
6154: SZ_CONTENTTYPE_HTML('String').SetString( SZ_CONTENTTYPE_HTMLA);
6155: SZ_CONTENTTYPE_CDF('String').SetString( 'application/x-cdf
6156: SZ_CONTENTTYPE_CDFW('String').SetString( 'application/x-cdf
6157: SZ_CONTENTTYPE_CDF('String').SetString( SZ_CONTENTTYPE_CDF);
6158: Function PathIsHTMLFile( pszPath : PChar) : BOOL
6159: STIF_DEFAULT('LongWord').SetUInt( $00000000);
6160: STIF_SUPPORT_HEX('LongWord').SetUInt( $00000001);
6161: Function StrNCmpI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer

```

```

6162: Function StrNCpy( psz1, psz2 : PChar; cchMax : Integer) : PChar
6163: Function StrCatN( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6164: Function PathAddBackslash( pszPath : PChar) : PChar
6165: Function PathAddExtension( pszPath : PChar; pszExt : PChar) : BOOL
6166: Function PathAppend( pszPath : PChar; pMore : PChar) : BOOL
6167: Function PathBuildRoot( szRoot : PChar; iDrive : Integer) : PChar
6168: Function PathCanonicalize( pszBuf : PChar; pszPath : PChar) : BOOL
6169: Function PathCombine( szDest : PChar; lpszDir, lpszFile : PChar) : PChar
6170: Function PathCompactPath( hDC : HDC; pszPath : PChar; dx : UINT) : BOOL
6171: Function PathCompactPathEx( pszOut : PChar; pszSrc : PChar; cchMax : UINT; dwFlags:DWORD) : BOOL
6172: Function PathCommonPrefix( pszFile1, pszFile2 : PChar; achPath : PChar) : Integer
6173: Function PathFileExists( pszPath : PChar) : BOOL
6174: Function PathFindExtension( pszPath : PChar) : PChar
6175: Function PathFindFileName( pszPath : PChar) : PChar
6176: Function PathFindNextComponent( pszPath : PChar) : PChar
6177: Function PathFindOnPath( pszPath : PChar; var ppszOtherDirs : PChar) : BOOL
6178: Function PathGetArgs( pszPath : PChar) : PChar
6179: Function PathFindSuffixArray( pszPath: PChar; const apszSuffix: PChar; iArraySize: Integer): PChar
6180: Function PathIsLNFFileSpec( lpName : PChar) : BOOL
6181: Function PathGetCharType( ch : Char) : UINT
6182: GCT_INVALID, 'LongWord').SetUInt( $0000);
6183: GCT_LFNCHAR, 'LongWord').SetUInt( $0001);
6184: GCT_SHORTCHAR, 'LongWord').SetUInt( $0002);
6185: GCT_WILD, 'LongWord').SetUInt( $0004);
6186: GCT_SEPARATOR, 'LongWord').SetUInt( $0008);
6187: Function PathGetDriveNumber( pszPath : PChar) : Integer
6188: Function PathIsDirectory( pszPath : PChar) : BOOL
6189: Function PathIsDirectoryEmpty( pszPath : PChar) : BOOL
6190: Function PathIsFileSpec( pszPath : PChar) : BOOL
6191: Function PathIsPrefix( pszPrefix, pszPath : PChar) : BOOL
6192: Function PathIsRelative( pszPath : PChar) : BOOL
6193: Function PathIsRoot( pszPath : PChar) : BOOL
6194: Function PathIsSameRoot( pszPath1, pszPath2 : PChar) : BOOL
6195: Function PathIsUNC( pszPath : PChar) : BOOL
6196: Function PathIsNetworkPath( pszPath : PChar) : BOOL
6197: Function PathIsUNCServer( pszPath : PChar) : BOOL
6198: Function PathIsUNCServerShare( pszPath : PChar) : BOOL
6199: Function PathIsContentType( pszPath, pszContentType : PChar) : BOOL
6200: Function PathIsURL( pszPath : PChar) : BOOL
6201: Function PathMakePretty( pszPath : PChar) : BOOL
6202: Function PathMatchSpec( pszFile, pszSpec : PChar) : BOOL
6203: Function PathParseIconLocation( pszIconFile : PChar) : Integer
6204: Procedure PathQuoteSpaces( lpsz : PChar)
6205: Function PathRelativePathTo( pszPath:PChar; pszFrom:PChar; dwAttrFrom:DWORD; pszTo:PChar; dwAttrTo:DWORD):BOOL;
6206: Procedure PathRemoveArgs( pszPath : PChar)
6207: Function PathRemoveBackslash( pszPath : PChar) : PChar
6208: Procedure PathRemoveBlanks( pszPath : PChar)
6209: Procedure PathRemoveExtension( pszPath : PChar)
6210: Function PathRemoveFileSpec( pszPath : PChar) : BOOL
6211: Function PathRenameExtension( pszPath : PChar; pszExt : PChar) : BOOL
6212: Function PathSearchAndQualify( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6213: Procedure PathSetDlgItemPath( hDlg : HWND; id : Integer; pszPath : PChar)
6214: Function PathSkipRoot( pszPath : PChar) : PChar
6215: Procedure PathStripPath( pszPath : PChar)
6216: Function PathStripToRoot( pszPath : PChar) : BOOL
6217: Procedure PathUnquoteSpaces( lpsz : PChar)
6218: Function PathMakeSystemFolder( pszPath : PChar) : BOOL
6219: Function PathUnmakeSystemFolder( pszPath : PChar) : BOOL
6220: Function PathIsSystemFolder( pszPath : PChar; dwAttrb : DWORD) : BOOL
6221: Procedure PathUndecorate( pszPath : PChar)
6222: Function PathUnExpandEnvStrings( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6223: URL_SCHEME_INVALID, 'LongInt').SetInt( - 1);
6224: URL_SCHEME_UNKNOWN, 'LongInt').SetInt( 0);
6225: URL_SCHEME_FTP, 'LongInt').SetInt( 1);
6226: URL_SCHEME_HTTP, 'LongInt').SetInt( 2);
6227: URL_SCHEME_GOPHER, 'LongInt').SetInt( 3);
6228: URL_SCHEME_MAILTO, 'LongInt').SetInt( 4);
6229: URL_SCHEME_NEWS, 'LongInt').SetInt( 5);
6230: URL_SCHEME_NNTP, 'LongInt').SetInt( 6);
6231: URL_SCHEME_TELNET, 'LongInt').SetInt( 7);
6232: URL_SCHEME_WAIS, 'LongInt').SetInt( 8);
6233: URL_SCHEME_FILE, 'LongInt').SetInt( 9);
6234: URL_SCHEME_MK, 'LongInt').SetInt( 10);
6235: URL_SCHEME_HTTPS, 'LongInt').SetInt( 11);
6236: URL_SCHEME_SHELL, 'LongInt').SetInt( 12);
6237: URL_SCHEME_SNEWS, 'LongInt').SetInt( 13);
6238: URL_SCHEME_LOCAL, 'LongInt').SetInt( 14);
6239: URL_SCHEME_JAVASCRIPT, 'LongInt').SetInt( 15);
6240: URL_SCHEME_VBSCRIPT, 'LongInt').SetInt( 16);
6241: URL_SCHEME_ABOUT, 'LongInt').SetInt( 17);
6242: URL_SCHEME_RES, 'LongInt').SetInt( 18);
6243: URL_SCHEME_MAXVALUE, 'LongInt').SetInt( 19);
6244: URL_SCHEME, 'Integer
6245: URL_PART_NONE, 'LongInt').SetInt( 0);
6246: URL_PART_SCHEME, 'LongInt').SetInt( 1);
6247: URL_PART_HOSTNAME, 'LongInt').SetInt( 2);
6248: URL_PART_USERNAME, 'LongInt').SetInt( 3);
6249: URL_PART_PASSWORD, 'LongInt').SetInt( 4);
6250: URL_PART_PORT, 'LongInt').SetInt( 5);

```



```

6251: URL_PART_QUERY', 'LongInt').SetInt( 6);
6252: URL_PART', 'DWORD
6253: URLIS_URL', 'LongInt').SetInt( 0);
6254: URLIS_OPAQUE', 'LongInt').SetInt( 1);
6255: URLIS_NOHISTORY', 'LongInt').SetInt( 2);
6256: URLIS_FILEURL', 'LongInt').SetInt( 3);
6257: URLIS_APPLIABLE', 'LongInt').SetInt( 4);
6258: URLIS_DIRECTORY', 'LongInt').SetInt( 5);
6259: URLIS_HASQUERY', 'LongInt').SetInt( 6);
6260: TUrlIs', 'DWORD
6261: URL_UNESCAPE', 'LongWord').SetUInt( $10000000);
6262: URL_ESCAPE_UNSAFE', 'LongWord').SetUInt( $20000000);
6263: URL_PLUGGABLE_PROTOCOL', 'LongWord').SetUInt( $40000000);
6264: URL_WININET_COMPATIBILITY', 'LongWord').SetUInt( DWORD ( $80000000 ));
6265: URL_DONT_ESCAPE_EXTRA_INFO', 'LongWord').SetUInt( $02000000);
6266: URL_ESCAPE_SPACES_ONLY', 'LongWord').SetUInt( $04000000);
6267: URL_DONT_SIMPLIFY', 'LongWord').SetUInt( $08000000);
6268: URL_NO_META', 'LongWord').SetUInt( URL_DONT_SIMPLIFY);
6269: URL_UNESCAPE_INPLACE', 'LongWord').SetUInt( $00100000);
6270: URL_CONVERT_IF_DOSPATH', 'LongWord').SetUInt( $00200000);
6271: URL_UNESCAPE_HIGH_ANSI_ONLY', 'LongWord').SetUInt( $00400000);
6272: URL_INTERNAL_PATH', 'LongWord').SetUInt( $00800000);
6273: URL_FILE_USE_PATHURL', 'LongWord').SetUInt( $00010000);
6274: URL_ESCAPE_PERCENT', 'LongWord').SetUInt( $00001000);
6275: URL_ESCAPE_SEGMENT_ONLY', 'LongWord').SetUInt( $00002000);
6276: URL_PARTFLAG_KEEPScheme', 'LongWord').SetUInt( $00000001);
6277: URL_APPLY_DEFAULT', 'LongWord').SetUInt( $00000001);
6278: URL_APPLY_GUESSScheme', 'LongWord').SetUInt( $00000002);
6279: URL_APPLY_GUESSFILE', 'LongWord').SetUInt( $00000004);
6280: URL_APPLY_FORCEAPPLY', 'LongWord').SetUInt( $00000008);
6281: Function UrlCompare( psz1, psz2 : PChar; fIgnoreSlash : BOOL ) : Integer
6282: Function UrlCombine(pszBase, pszRelative: PChar; pszCombin: PChar; out pcchCombin: DWORD; dwFlags: DWORD): HRESULT;
6283: Function UrlCanonicalize(pszUrl: PChar; pszCanonicalized: PChar; pcchCanonic: DWORD; dwFlags: DWORD): HRESULT;
6284: Function UrlIsOpaque( pszURL : PChar ) : BOOL
6285: Function UrlIsNoHistory( pszURL : PChar ) : BOOL
6286: Function UrlIsFileUrl( pszURL : PChar ) : BOOL
6287: Function UrlIs( pszUrl : PChar; UrlIs : TUrlIs ) : BOOL
6288: Function UrlGetLocation( psz1 : PChar ) : PChar
6289: Function UrlUnescape( pszUrl, pszUnescaped : PChar; pcchUnescaped: DWORD; dwFlags : DWORD ) : HRESULT
6290: Function UrlEscape(pszUrl: PChar; pszEscaped: PChar; pcchEscaped: DWORD; dwFlags : DWORD) : HRESULT
6291: Function UrlCreateFromPath(pszPath: PChar; pszUrl: PChar; pcchUrl: DWORD; dwFlags : DWORD) : HRESULT
6292: Function PathCreateFromUrl(pszUrl: PChar; pszPath: PChar; pcchPath: DWORD; dwFlags : DWORD) : HRESULT
6293: Function UrlHash( pszUrl : PChar; pbHash : BYTE; cbHash : DWORD ) : HRESULT
6294: Function UrlGetPart(pszIn: PChar; pszOut: PChar; pcchOut: DWORD; dwPart, dwFlags: DWORD) : HRESULT
6295: Function UrlApplyScheme( pszIn : PChar; pszOut : PChar; pcchOut : DWORD; dwFlags : DWORD) : HRESULT
6296: Function HashData( pbData : BYTE; cbData : DWORD; pbHash : BYTE; cbHash : DWORD) : HRESULT
6297: Function UrlEscapeSpaces( pszUrl : PChar; pszEscaped : PChar; pcchEscaped : DWORD) : HRESULT
6298: Function UrlUnescapeInPlace( pszUrl : PChar; dwFlags : DWORD) : HRESULT
6299: Function SHDeleteEmptyKey( hKey : HKEY; pszSubKey : PChar ) : DWORD
6300: Function SHDeleteKey( hKey : HKEY; pszSubKey : PChar ) : DWORD
6301: Function SHDeleteValue( hKey : HKEY; pszSubKey, pszValue : PChar ) : DWORD
6302: Function SHEnumKeyEx( hKey : HKEY; dwIndex : DWORD; pszName : PChar; var pcchName : DWORD) : Longint
6303: Function SHEnumValue( hKey : HKEY; dwIndex : DWORD; pszValueName: PChar; var
pcchValueName: DWORD; pdwType: DWORD; pvData : ___Pointer; pcbData : DWORD) : Longint
6304: Function SHQueryInfoKey(hKey: HKEY; pcSubKeys, pcchMaxSubKeyLen, pcVal, pcchMaxValueNameLen: DWORD): Longint;
6305: Function SHCopyKey( hkeySrc : HKEY; szSrcSubKey : PChar; hkeyDest : HKEY; fReserved : DWORD) : DWORD
6306: Function SHRegGetPath(hKey: HKEY; pcszSubKey, pcszValue: PChar; pszPath: PChar; dwFlags: DWORD): DWORD
6307: Function SHRegSetPath( hKey: HKEY; pcszSubKey, pcszValue, pcszPath : PChar; dwFlags : DWORD): DWORD
6308: SHREGDEL_DEFAULT', 'LongWord').SetUInt( $00000000);
6309: SHREGDEL_HKCU', 'LongWord').SetUInt( $00000001);
6310: SHREGDEL_HKLM', 'LongWord').SetUInt( $00000010);
6311: SHREGDEL_BOTH', 'LongWord').SetUInt( $00000011);
6312: SHREGENUM_DEFAULT', 'LongWord').SetUInt( $00000000);
6313: SHREGENUM_HKCU', 'LongWord').SetUInt( $00000001);
6314: SHREGENUM_HKLM', 'LongWord').SetUInt( $00000010);
6315: SHREGENUM_BOTH', 'LongWord').SetUInt( $00000011);
6316: SHREGSET_HKCU', 'LongWord').SetUInt( $00000001);
6317: SHREGSET_FORCE_HKCU', 'LongWord').SetUInt( $00000002);
6318: SHREGSET_HKLM', 'LongWord').SetUInt( $00000004);
6319: SHREGSET_FORCE_HKLM', 'LongWord').SetUInt( $00000008);
6320: TSHRegDelFlags', 'DWORD
6321: TSHRegEnumFlags', 'DWORD
6322: HUSKEY', 'THandle
6323: ASSOCF_INIT_NOREMAPCLSID', 'LongWord').SetUInt( $00000001);
6324: ASSOCF_INIT_BYEXENAME', 'LongWord').SetUInt( $00000002);
6325: ASSOCF_OPEN_BYEXENAME', 'LongWord').SetUInt( $00000002);
6326: ASSOCF_INIT_DEFAULTTOSTAR', 'LongWord').SetUInt( $00000004);
6327: ASSOCF_INIT_DEFAULTTOFOLDER', 'LongWord').SetUInt( $00000008);
6328: ASSOCF_NOUSERSETTINGS', 'LongWord').SetUInt( $00000010);
6329: ASSOCF_NOTRUNCATE', 'LongWord').SetUInt( $00000020);
6330: ASSOCF_VERIFY', 'LongWord').SetUInt( $00000040);
6331: ASSOCF_REMAPRUNDLL', 'LongWord').SetUInt( $00000080);
6332: ASSOCF_NOFIXUPS', 'LongWord').SetUInt( $00000100);
6333: ASSOCF_IGNOREBASECLASS', 'LongWord').SetUInt( $00000200);
6334: ASSOCF', 'DWORD
6335: ASSOCTSTR_COMMAND', 'LongInt').SetInt( 1);
6336: ASSOCTSTR_EXECUTABLE', 'LongInt').SetInt( 2);
6337: ASSOCTSTR_FRIENDLYDOCNAME', 'LongInt').SetInt( 3);
6338: ASSOCTSTR_FRIENDLYAPPNAME', 'LongInt').SetInt( 4);

```

```

6339: ASSOCSTR_NOOPEN', 'LongInt').SetInt( 5);
6340: ASSOCSTR_SHELLNEWVALUE', 'LongInt').SetInt( 6);
6341: ASSOCSTR_DDECOMMAND', 'LongInt').SetInt( 7);
6342: ASSOCSTR_DDEIFEXEC', 'LongInt').SetInt( 8);
6343: ASSOCSTR_DDEAPPLICATION', 'LongInt').SetInt( 9);
6344: ASSOCSTR_DDETOPIC', 'LongInt').SetInt( 10);
6345: ASSOCSTR_INFOTIP', 'LongInt').SetInt( 11);
6346: ASSOCSTR_MAX', 'LongInt').SetInt( 12);
6347: ASSOCSTR', 'DWORD
6348: ASSOCKEY_SHELLEXECCLASS', 'LongInt').SetInt( 1);
6349: ASSOCKEY_APP', 'LongInt').SetInt( 2);
6350: ASSOCKEY_CLASS', 'LongInt').SetInt( 3);
6351: ASSOCKEY_BASECLASS', 'LongInt').SetInt( 4);
6352: ASSOCKEY_MAX', 'LongInt').SetInt( 5);
6353: ASSOCKEY', 'DWORD
6354: ASSOCDATA_MSIDESCRIPTOR', 'LongInt').SetInt( 1);
6355: ASSOCDATA_NOACTIVATEHANDLER', 'LongInt').SetInt( 2);
6356: ASSOCDATA_QUERYCLASSSTORE', 'LongInt').SetInt( 3);
6357: ASSOCDATA_HASPERUSERASSOC', 'LongInt').SetInt( 4);
6358: ASSOCDATA_MAX', 'LongInt').SetInt( 5);
6359: ASSOCDATA', 'DWORD
6360: ASSOCENUM_NONE', 'LongInt').SetInt( 0);
6361: ASSOCENUM', 'DWORD
6362: SID_IQueryAssociations', 'String').SetString( '{c46ca590-3c3f-11d2-bee6-0000f805ca57}');
6363: SHACF_DEFAULT', 'LongWord').SetUInt( $00000000);
6364: SHACF_FILESYSTEM', 'LongWord').SetUInt( $00000001);
6365: SHACF_URLHISTORY', 'LongWord').SetUInt( $00000002);
6366: SHACF_URLMRU', 'LongWord').SetUInt( $00000004);
6367: SHACF_USETAB', 'LongWord').SetUInt( $00000008);
6368: SHACF_FILESYS_ONLY', 'LongWord').SetUInt( $00000010);
6369: SHACF_AUTOSUGGEST_FORCE_ON', 'LongWord').SetUInt( $10000000);
6370: SHACF_AUTOSUGGEST_FORCE_OFF', 'LongWord').SetUInt( $20000000);
6371: SHACF_AUTOAPPEND_FORCE_ON', 'LongWord').SetUInt( $40000000);
6372: SHACF_AUTOAPPEND_FORCE_OFF', 'LongWord').SetUInt( DWORD ( $80000000 ));
6373: Function SHAutoComplete( hwndEdit : HWND; dwFlags : DWORD ) : HRESULT
6374: Procedure SHSetThreadRef( punk : IUnknown)
6375: Procedure SHGetThreadRef( out ppunk : IUnknown)
6376: CTF_INSIST', 'LongWord').SetUInt( $00000001);
6377: CTF_THREAD_REF', 'LongWord').SetUInt( $00000002);
6378: CTF_PROCESS_REF', 'LongWord').SetUInt( $00000004);
6379: CTF_COINIT', 'LongWord').SetUInt( $00000008);
6380: Function SHCreateShellPalette( hdc : HDC ) : HPALETTE
6381: Procedure ColorRGBToHLS( clrRGB : TColorRef; out pwHue, pwLuminance, pwSaturation : WORD)
6382: Function ColorHLSToRGB( wHue, wLuminance, wSaturation : WORD) : TColorRef
6383: Function ColorAdjustLuma( clrRGB : TColorRef; n : Integer; fScale : Boolean) : TColorRef
6384: Function GetSysColorBrush( nIndex : Integer) : HBRUSH
6385: Function DrawFocusRect( hdc : HDC; const lprc : TRect) : BOOL
6386: Function FillRect( hdc : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6387: Function FrameRect( hdc : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6388: Function InvertRect( hdc : HDC; const lprc : TRect) : BOOL
6389: Function SetRect( var lprc : TRect; xLeft, yTop, xRight, yBottom : Integer) : BOOL
6390: Function SetRectEmpty( var lprc : TRect) : BOOL
6391: Function CopyRect( var lprcDst : TRect; const lprcSrc : TRect) : BOOL
6392: Function InflateRect( var lprc : TRect; dx, dy : Integer) : BOOL
6393: Function IntersectRect2( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6394: Function SubtractRect( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6395:
6396: Function InitializeFlatSB( hWnd : HWND) : Bool
6397: Procedure UninitializeFlatSB( hWnd : HWND)
6398: Function FlatSB_GetScrollProp( p1 : HWND; propIndex : Integer; p3 : PInteger) : Bool
6399: Function FlatSB_SetScrollProp( p1 : HWND; index : Integer; newValue : Integer; p4 : Bool) : Bool
6400: Function GET_APPCOMMAND_LPARAM( lParam : Integer) : Word //of JvWin32
6401: Function GET_DEVICE_LPARAM( lParam : Integer) : Word
6402: Function GET_MOUSEORKEY_LPARAM( lParam : Integer) : Integer
6403: Function GET_FLAGS_LPARAM( lParam : Integer) : Word
6404: Function GET_KEYSTATE_LPARAM( lParam : Integer) : Word
6405:
6406:
6407: // ***** 204 unit uPSI_ShellAPI;
6408: Function DragQueryFile( Drop : HDROP; FileIndex : UINT; FileName : PChar; cb : UINT) : UINT
6409: Function DragQueryPoint( Drop : HDROP; var Point : TPoint) : BOOL
6410: Procedure DragFinish( Drop : HDROP)
6411: Procedure DragAcceptFiles( Wnd : HWND; Accept : BOOL)
6412: Function ShellExecute( hWnd : HWND; Operation, FileName, Parameters, Directory : PChar; ShowCmd : Integer) : HINST
6413: Function FindExecutable( FileName, Directory : PChar; Result : PChar) : HINST
6414: Function ShellAbout( Wnd : HWND; szApp, szOtherStuff : PChar; Icon : HICON) : Integer
6415: Function DuplicateIcon( hInst : HINST; Icon : HICON) : HICON
6416: Function ExtractAssociatedIcon( hInst : HINST; lpIconPath : PChar; var lpiIcon : Word) : HICON
6417: Function ExtractIcon( hInst : HINST; lpszExeFileName : PChar; nIconIndex : UINT) : HICON
6418: Function SHAppBarMessage( dwMessage : DWORD; var pData : TAppBarData) : UINT
6419: Function DoEnvironmentSubst( szString : PChar; cbString : UINT) : DWORD
6420: Function ExtractIconEx( lpszFile : PChar; nIconIndex : Int; var phiconSmall : HICON; nIcons : UINT) : UINT;
6421: Procedure SHFreeNameMappings( hNameMappings : THandle)
6422:
6423: DLLVER_PLATFORM_WINDOWS', 'LongWord').SetUInt( $00000001);
6424: DLLVER_PLATFORM_NT', 'LongWord').SetUInt( $00000002);
6425: DLLVER_MAJOR_MASK', 'LongWord').SetUInt( Int64 ( $FFFF000000000000 ));
6426: DLLVER_MINOR_MASK', 'LongWord').SetUInt( Int64 ( $0000FFFF00000000 ));
6427: DLLVER_BUILD_MASK', 'LongWord').SetUInt( Int64 ( $00000000FFFF0000 ));

```

```

6428: DLLVER_QFE_MASK', 'LongWord').SetUInt( Int64 ( $000000000000FFFF ));
6429: Function MAKEDLLVERULL( Major, Minor, Build, Qfe : Word) : Int64
6430: Function SimpleXMLEncode( const S : string) : string
6431: Procedure SimpleXMLDecode( var S : string; TrimBlanks : Boolean)
6432: Function XMLEncode( const S : string) : string
6433: Function XMLDecode( const S : string) : string
6434: Function EntityEncode( const S : string) : string
6435: Function EntityDecode( const S : string) : string
6436:
6437: procedure RIRegister_CPort_Routines(S: TPSExec);
6438: Procedure EnumComPorts( Ports : TStrings)
6439: Procedure ListComPorts( Ports : TStrings)
6440: Procedure ComPorts( Ports : TStrings) //Alias to Arduino
6441: Function GetComPorts: TStringlist;
6442: Function StrToBaudRate( Str : string) : TBaudRate
6443: Function StrToStopBits( Str : string) : TStopBits
6444: Function StrToDataBits( Str : string) : TDataBits
6445: Function StrToParity( Str : string) : TParityBits
6446: Function StrToFlowControl( Str : string) : TFlowControl
6447: Function BaudRateToStr( BaudRate : TBaudRate) : string
6448: Function StopBitsToStr( StopBits : TStopBits) : string
6449: Function DataBitsToStr( DataBits : TDataBits) : string
6450: Function ParityToStr( Parity : TParityBits) : string
6451: Function FlowControlToStr( FlowControl : TFlowControl) : string
6452: Function ComErrorsToStr( Errors : TComErrors) : String
6453:
6454: Function GetMessage( var lpMsg : TMsg; hWnd : HWND; wParamFilterMin, wParamFilterMax : UInt) : BOOL
6455: Function DispatchMessage( const lpMsg : TMsg) : Longint
6456: Function TranslateMessage( const lpMsg : TMsg) : BOOL
6457: Function SetMessageQueue( cMessagesMax : Integer) : BOOL
6458: Function PeekMessage( var lpMsg: TMsg; hWnd: HWND; wParamFilterMin, wParamFilterMax, wRemoveMsg: UInt): BOOL
6459: Function GetMessagePos : DWORD
6460: Function GetMessageTime : Longint
6461: Function GetMessageExtraInfo : Longint
6462: Function GetSpecialFolderPath( const FolderName : string; CanCreate : Boolean) : string
6463: Procedure JAddToRecentDocs( const Filename : string)
6464: Procedure ClearRecentDocs
6465: Function ExtractIconFromFile( FileName : string; Index : Integer) : HICON
6466: Function CreateShellLink( const AppName, Desc : string; Dest : string) : string
6467: Procedure GetShellLinkInfo( const LinkFile : WideString; var SLI : TShellLinkInfo)
6468: Procedure SetShellLinkInfo( const LinkFile : WideString; const SLI : TShellLinkInfo)
6469: Function RecycleFile( FileToRecycle : string) : Boolean
6470: Function JCopyFile( FromFile, ToDir : string) : Boolean
6471: Function ShellObjectTypeEnumToConst( ShellObjectType : TShellObjectType) : UInt
6472: Function ShellObjectTypeConstToEnum( ShellObjectType : UInt) : TShellObjectType
6473: Function QueryServiceConfig2A( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
        DWORD; var pcbBytesNeeded : DWORD) : BOOL
6474: Function QueryServiceConfig2W( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
        DWORD; var pcbBytesNeeded : DWORD) : BOOL
6475: Function QueryServiceConfig2( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
        DWORD; var pcbBytesNeeded : DWORD) : BOOL
6476: Function EnumServicesStatusExA( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
        dwServiceState : DWORD; lpServices : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned,
        lpResumeHandle : DWORD; pszGroupName : LP
6477: Function EnumServicesStatusExW( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
        dwServiceState : DWORD; lpServices : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned,
        lpResumeHandle : DWORD; pszGroupNam
6478: Function EnumServicesStatusEx( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
        dwServiceState : DWORD; lpServices : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned,
        lpResumeHandle : DWORD; pszGroupName
6479: Function ConvertSidToStringSid( sid : PSID; var stringSid : LPWSTR) : BOOL
6480:
6481: ***** unit uPSI_JclPeImage;
6482:
6483: Function IsValidPeFile( const FileName : TFileName) : Boolean
6484: // Function PeGetNtHeaders( const FileName : TFileName; var NtHeaders : TImageNtHeaders) : Boolean
6485: Function PeCreateNameHintTable( const FileName : TFileName) : Boolean
6486: Function PeRebaseImage( const ImageName : TFileName; NewBase : DWORD; TimeStamp : DWORD; MaxNewSize :
        DWORD) : TJclRebaseImageInfo
6487: Function PeVerifyChecksum( const FileName : TFileName) : Boolean
6488: Function PeClearChecksum( const FileName : TFileName) : Boolean
6489: Function PeUpdateChecksum( const FileName : TFileName) : Boolean
6490: Function PeDoesExportFunction( const FileName : TFileName; const
        FuncName : string; Options : TJclSmartCompOptions) : Bool;
6491: Function PeIsExportFunctionForwardedEx( const FileName : TFileName; const FunctionName : string; var
        ForwardedName : string; Options : TJclSmartCompOptions) : Boolean
6492: Function PeIsExportFunctionForwarded( const FileName : TFileName; const
        FunctionName : string; Options : TJclSmartCompOptions) : Bool
6493: Function PeDoesImportFunction( const FileName : TFileName; const FunctionName : string; const LibraryName
        : string; Options : TJclSmartCompOptions) : Boolean
6494: Function PeDoesImportLibrary( const FileName : TFileName; const
        LibraryName : string; Recursive : Boolean) : Boolean;
6495: Function PeImportedLibraries( const FileName : TFileName; const LibrariesList : TStrings; Recursive :
        Boolean; FullPathName : Boolean) : Boolean
6496: Function PeImportedFunctions( const FileName : TFileName; const FunctionsList : TStrings; const
        LibraryName : string; IncludeLibNames : Boolean) : Boolean
6497: Function PeExportedFunctions( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6498: Function PeExportedNames( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6499: Function PeExportedVariables( const FileName : TFileName; const FunctionsList : TStrings) : Boolean

```

```

6500: Function PeResourceKindNames(const FileN:TFileName;ResourceType:TJclPeResourceKind;const
NamesList:TStrings):Bool
6501: Function PeBorFormNames( const FileName : TFileName; const NamesList : TStrings) : Boolean
6502: Function PeBorDependedPackages(const FileName:TFileName;PackagesList:TStrings;FullPathName,
Descript:Bool):Bool;
6503: Function PeFindMissingImports( const FileName : TFileName; MissingImportsList : TStrings) : Boolean;
6504: Function PeFindMissingImports1( RequiredImportsList, MissingImportsList : TStrings) : Boolean;
6505: Function PeCreateRequiredImportList(const FileName: TFileName; RequiredImportsList: TStrings): Boolean;
6506: //Function PeMapImgNtHeaders( const BaseAddress : Pointer) : PImageNtHeaders
6507: //Function PeMapImgLibraryName( const BaseAddress : Pointer) : string
6508: //Function PeMapImgSections( const NtHeaders : PImageNtHeaders) : PImageSectionHeader
6509: //Function PeMapImgFindSection( const NtHeaders : PImageNtHeaders; const SectionName : string) :
PImageSectionHeader
6510: //Function PeMapImgExportedVariables(const Module: HMODULE; const VariablesList:TStrings):Boolean
6511: //Function PeMapImgResolvePackageThunk( Address : Pointer) : Pointer
6512: Function PeMapFindResource(const Module:HMODULE;const ResourceType:PChar;const ResourceName:string):
____Pointer;
6513: SIRegister_TJclPeSectionStream(CL);
6514: SIRegister_TJclPeMapImgHookItem(CL);
6515: SIRegister_TJclPeMapImgHooks(CL);
6516: //Function PeDbgImgNtHeaders(ProcessHandle:THandle;BaseAddress:Pointer;var
NtHeaders:TImageNtHeaders):Boolean
6517: //Function PeDbgImgLibraryName(ProcessHandle:THandle; BaseAddress:Pointer; var Name:string):Boolean
6518: Type TJclBorUmSymbolKind','(skData,skFunction,skConstructor,skDestructor,skRTTI,skVTable)
6519: TJclBorUmSymbolModifier','( smQualified, smLinkProc )
6520: TJclBorUmSymbolModifiers',' 'set of TJclBorUmSymbolModifier
6521: TJclBorUmDescription',' 'record Kind : TJclBorUmSymbolKind; Modifiers : TJclBorUmSymbolModifiers; end
6522: TJclBorUmResult',' ( urOk, urNotMangled, urMicrosoft, urError )
6523: TJclPeUmResult',' ( umNotMangled, umBorland, umMicrosoft )
6524: Function PeBorUnmangleName( const Name : string; var Unmangled : string; var Description :
TJclBorUmDescription; var BasePos : Integer) : TJclBorUmResult;
6525: Function PeBorUnmangleName1(const Name:string;var Unmangled:string;var
Descript:TJclBorUmDescription):TJclBorUmResult;
6526: Function PeBorUnmangleName2( const Name : string; var Unmangled : string) : TJclBorUmResult;
6527: Function PeBorUnmangleName3( const Name : string) : string;
6528: Function PeIsNameMangled( const Name : string) : TJclPeUmResult
6529: Function PeUnmangleName( const Name : string; var Unmangled : string) : TJclPeUmResult
6530:
6531:
6532: //***** SysTools uPSI_StSystem; *****
6533: Function StCopyFile( const SrcPath, DestPath : AnsiString) : Cardinal
6534: Function CreateTempFile( const aFolder : AnsiString; const aPrefix : AnsiString) : AnsiString
6535: Function DeleteVolumeLabel( Drive : Char) : Cardinal
6536: //Procedure EnumerateDirectories(const
StartDir:AnsiString;FL:TStrings;SubDirs:Bool;IncludeItem:TIncludeItemFunc);
6537: //Procedure EnumerateFiles(const StartDir:AnsiString;FL:TStrings;SubDirs:Bool
IncludeItem:TIncludeItemFunc);
6538: Function FileHandlesLeft( MaxHandles : Cardinal) : Cardinal
6539: Function FileMatchesMask( const FileName, FileMask : AnsiString) : Boolean
6540: Function FileTimeToStDateTime( FileTime : LongInt) : TStDateTimeRec
6541: Function FindNthSlash( const Path : AnsiString; n : Integer) : Integer
6542: Function FlushOsBuffers( Handle : Integer) : Boolean
6543: Function GetCurrentUser : AnsiString
6544: Function GetDiskClass( Drive : Char) : DiskClass
6545: Function GetDiskInfo(Drive:Char;var ClustersAvail,TotalClusters,BytesPerSector,
SectorsPerCluster:Cardinal):Bool;
6546: Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Double;var TotalSpaceAvail:Double;var
DiskSize:Double):Bool;
6547: Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Comp;var TotalSpaceAvail:Comp;var
DiskSize:Comp):Boolean;
6548: { index 0 - FreeBytesAvailable, 1 - TotalNumberOfBytes, 2 - TotalNumberOfFreeBytes }
6549: Function getDiskSpace2(const path: String; index: integer): int64;
6550: Function GetFileCreateDate( const FileName : AnsiString) : TDateTime
6551: Function StGetFileLastAccess( const FileName : AnsiString) : TDateTime
6552: Function GetFileLastModify( const FileName : AnsiString) : TDateTime
6553: Function GetHomeFolder( aForceSlash : Boolean) : AnsiString
6554: Function GetLongPath( const APath : AnsiString) : AnsiString
6555: Function GetMachineName : AnsiString
6556: Function GetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6557: Function GetParentFolder( const APath : AnsiString; aForceSlash : Boolean) : AnsiString
6558: Function GetShortPath( const APath : AnsiString) : AnsiString
6559: Function GetSystemFolder( aForceSlash : Boolean) : AnsiString
6560: Function GetTempFolder( aForceSlash : boolean) : AnsiString
6561: Function StGetWindowsFolder( aForceSlash : boolean) : AnsiString
6562: Function GetWorkingFolder( aForceSlash : boolean) : AnsiString
6563: Function GlobalDateTimeToLocal( const UTC : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6564: Function StIsDirectory( const DirName : AnsiString) : Boolean
6565: Function IsDirectoryEmpty( const S : AnsiString) : Integer
6566: Function IsDriveReady( Drive : Char) : Boolean
6567: Function IsFile( const FileName : AnsiString) : Boolean
6568: Function IsFileArchive( const S : AnsiString) : Integer
6569: Function IsFileHidden( const S : AnsiString) : Integer
6570: Function IsFileReadOnly( const S : AnsiString) : Integer
6571: Function IsFileSystem( const S : AnsiString) : Integer
6572: Function LocalDateTimeToGlobal( const DT1 : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6573: Function ReadVolumeLabel( var VolName : AnsiString; Drive : Char) : Cardinal
6574: Function SameFile( const FilePath1, FilePath2 : AnsiString; var ErrorCode : Integer) : Boolean
6575: Function SetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6576: Procedure SplitPath( const APath : AnsiString; Parts : TStrings)

```



```

6577: Function StDateTimeToFileTime( const FileTime : TStDateTimeRec ) : LongInt
6578: Function StDateTimeToUnixTime( const DT1 : TStDateTimeRec ) : Longint
6579: Function UnixTimeToStDateTime( UnixTime : Longint ) : TStDateTimeRec
6580: Function ValidDrive( Drive : Char ) : Boolean
6581: Function WriteVolumeLabel( const VolName : AnsiString; Drive : Char ) : Cardinal
6582:
6583: //*****unit uPSI_JellANMan;*****
6584: Function CreateAccount( const Server, Username, Fullname, Password, Description, Homedir, Script : string;
const PasswordNeverExpires : Boolean ) : Boolean
6585: Function CreateLocalAccount( const Username, Fullname, Password, Description, Homedir, Script : string;
const PasswordNeverExpires : Boolean ) : Boolean
6586: Function DeleteAccount( const Servername, Username : string ) : Boolean
6587: Function DeleteLocalAccount( Username : string ) : Boolean
6588: Function CreateLocalGroup( const Server, Groupname, Description : string ) : Boolean
6589: Function CreateGlobalGroup( const Server, Groupname, Description : string ) : Boolean
6590: Function DeleteLocalGroup( const Server, Groupname : string ) : Boolean
6591: Function GetLocalGroups( const Server : string; const Groups : TStrings ) : Boolean
6592: Function GetGlobalGroups( const Server : string; const Groups : TStrings ) : Boolean
6593: Function LocalGroupExists( const Group : string ) : Boolean
6594: Function GlobalGroupExists( const Server, Group : string ) : Boolean
6595: Function AddAccountToLocalGroup( const Accountname, Groupname : string ) : Boolean
6596: Function LookupGroupName( const Server : string; const RID : TNetWellKnownRID ) : string
6597: Procedure ParseAccountName( const QualifiedName : string; var Domain, UserName : string )
6598: Function IsLocalAccount( const AccountName : string ) : Boolean
6599: Function TimeStampInterval( StartStamp, EndStamp : TDateTime ) : integer
6600: Function GetRandomString( NumChar : cardinal ) : string
6601:
6602: //*****unit uPSI_CUtils;*****
6603: TypeS('TUnitType', '( utSrc, utHead, utRes, utPrj, utOther )
6604: Function cIsWinNT : boolean
6605: Procedure cFilesFromWildcard(Directory,Mask : string;var Files:TStringList;Subdirs,ShowDirs,
Multitasking:Boolean;
6606: Function cExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer ) : THandle
6607: Function cRunAndGetOutput( Cmd,WorkDir:string; ErrFunc:TErrFunc; LineOutputFunc:TLineOutputFunc;
CheckAbortFunc : TCheckAbortFunc; ShowReturnValue : Boolean ) : string
6608: Function cGetShortName( FileName : string ) : string
6609: Procedure cShowError( Msg : String )
6610: Function cCommaStrToStr( s : string; formatstr : string ) : string
6611: Function cIncludeQuoteIfSpaces( s : string ) : string
6612: Function cIncludeQuoteIfNeeded( s : string ) : string
6613: Procedure cLoadFilefromResource( const FileName : string; ms : TMemoryStream )
6614: Function cValidateFile(const FileName:string; const WorkPath:string;const CheckDirs:boolean):string;
6615: Function cBuildFilter( var value : string; const FLTStyle : TFILTERSET ) : boolean;
6616: Function cBuildFilter1( var value : string; const _filters : array of string ) : boolean;
6617: Function cCodeInstoStr( s : string ) : string
6618: Function cStrtoCodeIns( s : string ) : string
6619: Procedure cStrtoAttr( var Attr : TSynHighlighterAttributes; Value : string )
6620: Function cAttrtoStr( const Attr : TSynHighlighterAttributes ) : string
6621: Procedure cStrtoPoint( var pt : TPoint; value : string )
6622: Function cPointtoStr( const pt : TPoint ) : string
6623: Function cListtoStr( const List : TStrings ) : string
6624: Function ListtoStr( const List : TStrings ) : string
6625: Procedure StrtoList( s : string; const List : TStrings; const delimiter : char )
6626: Procedure cStrtoList( s : string; const List : TStrings; const delimiter : char )
6627: Function cGetFileType( const FileName : string ) : TUnitType
6628: Function cGetExTyp( const FileName : string ) : TExUnitType
6629: Procedure cSetPath( Add : string; const UseOriginal : boolean )
6630: Function cExpandFileto( const FileName : string; const BasePath : string ) : string
6631: Function cFileSamePath( const FileName : string; const TestPath : string ) : boolean
6632: Procedure cCloneMenu( const FromMenu : TMenuItem; ToMenu : TMenuItem )
6633: Function cGetLastPos( const SubStr : string; const S : string ) : integer
6634: Function cGenMakePath( FileName : String ) : String;
6635: Function cGenMakePath2( FileName : String ) : String
6636: Function cGenMakePath1( FileName : String; EscapeSpaces, EncloseInQuotes : Boolean ) : String;
6637: Function cGetRealPath( BrokenFileName : String; Directory : String ) : String
6638: Function cCalcMod( Count : Integer ) : Integer
6639: Function cGetVersionString( FileName : string ) : string
6640: Function cCheckChangeDir( var Dir : string ) : boolean
6641: Function cGetAssociatedProgram(const Extension:string; var Filename,Description: string):boolean
6642: Function cIsNumeric( s : string ) : boolean
6643: Procedure StrtoAttr( var Attr : TSynHighlighterAttributes; Value : string )
6644: Function AttrtoStr( const Attr : TSynHighlighterAttributes ) : string
6645: Function cGetFileType( const FileName : string ) : TUnitType
6646: Function Atoi(const aStr: string): integer
6647: Function Itoa(const aint: integer): string
6648:
6649:
6650: procedure SIRegister_cHTTPUtils(CL: TPSPascalCompiler);
6651: begin
6652:   FindClass('TOBJECT'),'EHTTP
6653:   FindClass('TOBJECT'),'EHTTPParser
6654:   //AnsiCharSet', 'set of AnsiChar
6655:   AnsiStringArray', 'array of AnsiString
6656:   THHTTPProtocolEnum', '( hpNone, hpCustom, hpHTTP, hpHTTPS )
6657:   THHTTPVersionEnum', '( hvNone, hvCustom, hvHTTP10, hvHTTP11 )
6658:   THHTTPVersion', 'record Version : THHTTPVersionEnum; Protocol : TH
6659:   + 'THHTTPProtocolEnum; CustomProtocol : AnsiString; CustomMajVersion : Integer; '
6660:   + 'CustomMinVersion : Integer; end
6661:   THHTTPHeaderNameEnum', '( hntCustom, hntHost, hntContentType, hnt'

```

```

6662: + 'ContentLength, hntContentTransferEncoding, hntContentLocation, hntContentL'
6663: + 'anguage, hntContentEncoding, hntTransferEncoding, hntDate, hntServer, hntU'
6664: + 'serAgent, hntLocation, hntConnection, hntExpires, hntCacheControl, hntSetC'
6665: + 'ookie, hntCookie, hntAuthorization, hntVia, hntWarning, hntContentRange, h'
6666: + 'ntXForwardedFor, hntPragma, hntXPoweredBy, hntWWWAuthenticate, hntLastModi'
6667: + 'fied, hntETag, hntProxyAuthorization, hntReferer, hntAge, hntAcceptRanges,'
6668: + ' hntAcceptEncoding, hntAcceptLanguage, hntAcceptCharset, hntIfModifiedSinc'
6669: + 'e, hntIfUnmodifiedSince, hntRetryAfter, hntUpgrade, hntStatus, hntProxyCon'
6670: + 'nection, hntOrigin, hntKeepAlive )
6671: THTTPHeaderName', 'record Value : THTTPHeaderNameEnum; Custom: AnsiString; end
6672: THTTPCustomHeader', 'record FieldName: AnsiString; FieldValue : '
6673: + ' AnsiString; end
6674: //PHTTPCustomHeader', '^THTTPCustomHeader // will not work
6675: THTTPContentLengthEnum', '( hcltNone, hcltByteCount )
6676: THTTPContentLength', 'record Value : THTTPContentLengthEnum; ByteCount: Int64; end
6677: //PHTTPContentLength', '^THTTPContentLength // will not work
6678: THTTPContentTypeMajor', '( hctmCustom, hctmText, hctmImage )
6679: THTTPContentTypeEnum', '( hctNone, hctCustomParts, hctCustomStri'
6680: + 'ng, hctTextHtml, hctTextAscii, hctTextCss, hctTextPlain, hctTextXml, hctTe'
6681: + 'xtCustom, hctImageJpeg, hctImagePng, hctImageGif, hctImageCustom, hctAppli'
6682: + 'cationJSON, hctApplicationOctetStream, hctApplicationJavaScript, hctApplic'
6683: + 'ationCustom, hctAudioCustom, hctVideoCustom )
6684: THTTPContentType', 'record Value : THTTPContentTypeEnum; CustomM'
6685: + 'ajor : AnsiString; CustomMinor : AnsiString; Parameters : AnsiStringArray;'
6686: + ' CustomStr : AnsiString; end
6687: THTTPDateFieldEnum', '( hdNone, hdCustom, hdParts, hdDateTime )
6688: THTTPDateField', 'record Value : THTTPDateFieldEnum; DayOfWeek : '
6689: + ' Integer; Day : integer; Month : integer; Year : Integer; Hour : integer; '
6690: + 'Min : integer; Sec : Integer; TimeZoneGMT : Boolean; CustomTimeZone : Ansi'
6691: + 'String; DateTime : TDateTime; Custom : AnsiString; end
6692: THTTPTransferEncodingEnum', '( hteNone, hteCustom, hteChunked )
6693: THTTPTransferEncoding', 'record Value : THTTPTransferEncodingEnum'
6694: + 'm; Custom : AnsiString; end
6695: THTTPConnectionFieldEnum', '( hcfNone, hcfCustom, hcfClose, hcfKeepAlive )
6696: THTTPConnectionField', 'record Value : THTTPConnectionFieldEnum;'
6697: + ' Custom : AnsiString; end
6698: THTTPPageFieldEnum', '( hafNone, hafCustom, hafAge )
6699: THTTPPageField', 'record Value: THTTPPageFieldEnum; Age : Int64; Custom: AnsiString; end
6700: THTTPCacheControlFieldEnum', '( hccfNone, hccfDecoded, hccfCustom )
6701: THTTPCacheControlRequestSubField', '( hccsfNoCache, hccsfNoStore'
6702: + ', hccsfMaxAge, hccsfMaxStale, hccsfMinFresh, hccsfNoTransform, hccsfOnlyIfCached )
6703: THTTPCacheControlResponseSubField', '( hccrfPublic, hccrfPrivate'
6704: + ', hccrfNoCache, hccrfNoStore, hccrfNoTransform, hccrfMustRevalidate, hccrf'
6705: + 'ProxyRevalidate, hccrfMaxAge, hccrfSMaxAge )
6706: THTTPCacheControlField', 'record Value : THTTPCacheControlFieldEnum; end
6707: THTTPContentEncodingEnum', '( hceNone, hceCustom, hceIdentity, h'
6708: + 'ceCompress, hceDeflate, hceExi, hceGzip, hcePack200Gzip )
6709: THTTPContentEncoding', 'record Value: THTTPContentEncodingEnum; Custom: AnsiString; end;
6710: THTTPContentEncodingFieldEnum', '( hcefNone, hcefList )
6711: THTTPContentEncodingField', 'record Value : THTTPContentEncoding'
6712: + 'FieldEnum; List : array of THTTPContentEncoding; end
6713: THTTPRetryAfterFieldEnum', '( hrafNone, hrafCustom, harfDate, harfSeconds )
6714: THTTPRetryAfterField', 'record Value : THTTPRetryAfterFieldEnum;'
6715: + ' Custom : AnsiString; Date : TDateTime; Seconds : Int64; end
6716: THTTPContentRangeFieldEnum', '( hcrfNone, hcrfCustom, hcrfByteRange )
6717: THTTPContentRangeField', 'record Value : THTTPContentRangeFieldE'
6718: + 'num; ByteFirst : Int64; ByteLast : Int64; ByteSize : Int64; Custom : AnsiString; end
6719: THTTPSetCookieFieldEnum', '( hskoNone, hskoDecoded, hskoCustom )
6720: THTTPSetCookieCustomField', 'record Name : AnsiString; Value : AnsiString; end
6721: THTTPSetCookieCustomFieldArray', 'array of THTTPSetCookieCustomField
6722: THTTPSetCookieField', 'record Value : THTTPSetCookieFieldEnum; D'
6723: + 'omain : AnsiString; Path : AnsiString; Expires : THTTPDateField; MaxAge : '
6724: + 'Int64; HttpOnly : Boolean; Secure : Boolean; CustomFields : THTTPSetCookie'
6725: + 'CustomFieldArray; Custom : AnsiString; end
6726: //PHTTPSetCookieField', '^THTTPSetCookieField // will not work
6727: THTTPSetCookieFieldArray', 'array of THTTPSetCookieField
6728: THTTPCookieFieldEnum', '( hcoNone, hcoDecoded, hcoCustom )
6729: THTTPCookieFieldEntry', 'record Name : AnsiString; Value : AnsiString; end
6730: //PHTTPCookieFieldEntry', '^THTTPCookieFieldEntry // will not work
6731: THTTPCookieFieldEntryArray', 'array of THTTPCookieFieldEntry
6732: THTTPCookieField', 'record Value : THTTPCookieFieldEnum; Entries'
6733: + ' : THTTPCookieFieldEntryArray; Custom : AnsiString; end
6734: THTTPCommonHeaders', 'record TransferEncoding : THTTPTransferEnc'
6735: + 'oding; ContentType : THTTPContentType; ContentLength : THTTPContentLength;'
6736: + ' Connection : THTTPConnectionField; ProxyConnection : THTTPConnectionField'
6737: + '; Date : THTTPDateField; ContentEncoding : THTTPContentEncodingField; end
6738: THTTPCustomHeaders', 'array of THTTPCustomHeader
6739: //THTTFFixedHeaders', 'array[THTTPHeaderNameEnum] of AnsiString
6740: THTTFFixedHeaders', 'array[0..42] of AnsiString
6741: THTTPMethodEnum', '( hmNone, hmCustom, hmGET, hmPUT, hmPOST, hmC'
6742: + 'ONNECT, hmHEAD, hmDELETE, hmOPTIONS, hmTRACE )
6743: THTTPMethod', 'record Value : THTTPMethodEnum; Custom : AnsiString; end
6744: THTTPRequestStartLine', 'record Method: THTTPMethod; URI: AnsiString; Version: THTTPVersion; end
6745: THTTPRequestHeader', 'record CommonHeaders : THTTPCommonHeaders;'
6746: + ' FixedHeaders : THTTFFixedHeaders; CustomHeaders : THTTPCustomHeaders; Coo'
6747: + 'kie : THTTPCookieField; IfModifiedSince: THTTPDateField; IfUnmodifiedSince: THTTPDateField; end
6748: //PHTTPRequestHeader', '^THTTPRequestHeader // will not work
6749: THTTPRequest', 'record StartLine : THTTPRequestStartLine; Header'
6750: + ' : THTTPRequestHeader; HeaderComplete : Boolean; HasContent : Boolean; end

```

```

6751:   THTTPResponseStartLineMessage', '( hslmNone, hslmCustom, hslmOK)
6752:   THTTPResponseStartLine', 'record Version : THTTPVersion; Code : '
6753:   + 'Integer; Msg : THTTPResponseStartLineMessage; CustomMsg : AnsiString; end
6754:   THTTPResponseHeader', 'record CommonHeaders : THTTPCommonHeaders'
6755:   + 'FixedHeaders : THTTFFixedHeaders; CustomHeaders : THTTPCustomHeaders; Co'
6756:   + 'okies : THTTPSetCookieFieldArray; Expires : THTTPOdateField; LastModified : '
6757:   + 'THTTPOdateField; Age : THTTPOageField; end
6758:   //PHTTPOResponseHeader', 'THTTPOResponseHeader // will not work
6759:   THTTPResponse', 'record StartLine : THTTPResponseStartLine; Head'
6760:   + 'er : THTTPOResponseHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6761:   Function HTTPMessageHasContent( const H : THTTPCommonHeaders ) : Boolean
6762:   Procedure InitHTTPRequest( var A : THTTPRequest)
6763:   Procedure InitHTTPResponse( var A : THTTPResponse)
6764:   Procedure ClearHTTPVersion( var A : THTTPVersion)
6765:   Procedure ClearHTTPContentLength( var A : THTTPContentLength)
6766:   Procedure ClearHTTPContentType( var A : THTTPContentType)
6767:   Procedure ClearHTTPDateField( var A : THTTPOdateField)
6768:   Procedure ClearHTTPTransferEncoding( var A : THTTPOTransferEncoding)
6769:   Procedure ClearHTTPConnectionField( var A : THTTPOConnectionField)
6770:   Procedure ClearHTTPOageField( var A : THTTPOageField)
6771:   Procedure ClearHTTPContentEncoding( var A : THTTPOContentEncoding)
6772:   Procedure ClearHTTPContentEncodingField( var A : THTTPOContentEncodingField)
6773:   Procedure ClearHTTPContentRangeField( var A : THTTPOContentRangeField)
6774:   Procedure ClearHTTPSetCookieField( var A : THTTPOSetCookieField)
6775:   Procedure ClearHTTPCommonHeaders( var A : THTTPCommonHeaders)
6776:   //Procedure ClearHTTPOFixedHeaders( var A : THTTPOFixedHeaders)
6777:   Procedure ClearHTTPOCustomHeaders( var A : THTTPOCustomHeaders)
6778:   Procedure ClearHTTPOCookieField( var A : THTTPOCookieField)
6779:   Procedure ClearHTTPMethod( var A : THTTPOMethod)
6780:   Procedure ClearHTTPRequestStartLine( var A : THTTPRequestStartLine)
6781:   Procedure ClearHTTPRequestHeader( var A : THTTPRequestHeader)
6782:   Procedure ClearHTTPRequest( var A : THTTPRequest)
6783:   Procedure ClearHTTPResponseStartLine( var A : THTTPResponseStartLine)
6784:   Procedure ClearHTTPResponseHeader( var A : THTTPOResponseHeader)
6785:   Procedure ClearHTTPResponse( var A : THTTPResponse)
6786:   THTTPOStringOption', '( hsoNone )
6787:   THTTPOStringOptions', 'set of THTTPOStringOption
6788:   FindClass( 'TOBJECT', 'TAnsiStringBuilder
6789:
6790:   Procedure BuildStrHTTPVersion(const A:THTTPVersion; const B:TAnsiStringBuilder;const
P:THTTPOStringOptions);
6791:   Procedure BuildStrHTTPContentLengthValue(const A: THTTPContentLength; const B : TAnsiStringBuilder; const
P : THTTPOStringOptions)
6792:   Procedure BuildStrHTTPContentLength( const A : THTTPContentLength; const B : TAnsiStringBuilder; const P
: THTTPOStringOptions)
6793:   Procedure BuildStrHTTPContentTypeValue( const A : THTTPContentType; const B : TAnsiStringBuilder; const P
: THTTPOStringOptions)
6794:   Procedure BuildStrHTTPContentType(const A:THTTPContentType;const B:TAnsiStringBuilder; const
P:THTTPOStringOptions)
6795:   Procedure BuildStrRFCDateTime( const DOW, Da, Mo, Ye, Ho, Mi, Se : Integer; const TZ : AnsiString; const
B : TAnsiStringBuilder; const P : THTTPOStringOptions)
6796:   Procedure BuildStrHTTPOdateFieldValue( const A : THTTPOdateField; const B : TAnsiStringBuilder; const P :
THTTPOStringOptions)
6797:   Procedure BuildStrHTTPOdateField(const A:THTTPOdateField;const B:TAnsiStringBuilder;const
P:THTTPOStringOptions);
6798:   Procedure BuildStrHTTPTransferEncodingValue( const A : THTTPOTransferEncoding; const B :
TAnsiStringBuilder; const P : THTTPOStringOptions)
6799:   Procedure BuildStrHTTPTransferEncoding( const A : THTTPOTransferEncoding; const B : TAnsiStringBuilder;
const P : THTTPOStringOptions)
6800:   Procedure BuildStrHTTPContentRangeField( const A : THTTPOContentRangeField; const B : TAnsiStringBuilder;
const P : THTTPOStringOptions)
6801:   Procedure BuildStrHTTPConnectionFieldValue( const A : THTTPOConnectionField; const B : TAnsiStringBuilder;
const P : THTTPOStringOptions)
6802:   Procedure BuildStrHTTPConnectionField( const A : THTTPOConnectionField; const B : TAnsiStringBuilder;
const P : THTTPOStringOptions)
6803:   Procedure BuildStrHTTPOageField(const A:THTTPOageField;const B:TAnsiStringBuilder;const
P:THTTPOStringOptions);
6804:   Procedure BuildStrHTTPContentEncoding( const A : THTTPOContentEncoding; const B : TAnsiStringBuilder;
const P : THTTPOStringOptions)
6805:   Procedure BuildStrHTTPContentEncodingField(const A:THTTPOContentEncodingField;const
B:TAnsiStringBuilder;const P:THTTPOStringOptions)
6806:   Procedure BuildStrHTTPProxyConnectionField(const A : THTTPOConnectionField; const B : TAnsiStringBuilder;
const P : THTTPOStringOptions)
6807:   Procedure BuildStrHTTPCommonHeaders( const A : THTTPCommonHeaders; const B : TAnsiStringBuilder; const P
: THTTPOStringOptions)
6808:   Procedure BuildStrHTTPOFixedHeaders(const A:THTTPOFixedHeaders;const B:TAnsiStrBuilder;const
P:THTTPOStringOptions)
6809:   Procedure BuildStrHTTPOCustomHeaders( const A : THTTPOCustomHeaders; const B : TAnsiStringBuilder; const P
: THTTPOStringOptions)
6810:   Procedure BuildStrHTTPSetCookieFieldValue( const A : THTTPOSetCookieField; const B : TAnsiStringBuilder;
const P : THTTPOStringOptions)
6811:   Procedure BuildStrHTTPOCookieFieldValue( const A : THTTPOCookieField; const B : TAnsiStringBuilder; const P
: THTTPOStringOptions)
6812:   Procedure BuildStrHTTPOCookieField(const A:THTTPOCookieField;const B:TAnsiStringBuilder;const
P:THTTPOStringOptions);
6813:   Procedure BuildStrHTTPMethod( const A : THTTPOMethod; const B : TAnsiStringBuilder; const P :
THTTPOStringOptions)
6814:   Procedure BuildStrHTTPRequestStartLine( const A : THTTPRequestStartLine; const B : TAnsiStringBuilder;
const P : THTTPOStringOptions)

```

```

6815: Procedure BuildStrHTTPRequestHeader(const A:THTTPRequestHeader;const B:TAnsiStringBuilder;const
P:THTTPStringOptions);
6816: Procedure BuildStrHTTPRequest(const A : THTTPRequest; const B : TAnsiStringBuilder; const P :
THTTPStringOptions)
6817: Procedure BuildStrHTTPResponseCookieFieldArray(const A : THTTPSetCookieFieldArray; const B :
TAnsiStringBuilder; const P : THTTPStringOptions)
6818: Procedure BuildStrHTTPResponseStartLine(const A:THTTPResponseStartLine;const B:TAnsiStrBldr;const P
THTTPStrOptions);
6819: Procedure BuildStrHTTPResponseHeader(const A:THTTPRespHeader;const B:TAnsiStrBuilder;const
P:THTTPStringOptions);
6820: Procedure BuildStrHTTPResponse(const A:THTTPResponse; const B:TAnsiStringBuilder; const
P:THTTPStringOptions);
6821: Function HTTPContentTypeValueToStr(const A : THTTPContentType) : AnsiString
6822: Function HTTPSetCookieFieldValueToStr(const A : THTTPSetCookieField) : AnsiString
6823: Function HTTPCookieFieldValueToStr(const A : THTTPCookieField) : AnsiString
6824: Function HTTPMethodToStr(const A : THTTPMethod) : AnsiString
6825: Function HTTPRequestToStr(const A : THTTPRequest) : AnsiString
6826: Function HTTPResponseToStr(const A : THTTPResponse) : AnsiString
6827: Procedure PrepareCookie(var A:THTTPCookieField;const B:THTTPSetCookieFieldArray;const
Domain:AnsiString;const Secure:Bool; THTTPParserHeaderParseFunc', 'Function ( const HeaderName : THTT
6828: +PHeaderNameEnum; const HeaderPtr : __Pointer) : Boolean
6829: SIRegister_THTTPParser(CL);
6830: FindClass('TOBJECT'),'THTTPContentDecoder
6831: THTTPContentDecoderProc', 'Procedure ( const Sender : THTTPContentDecoder)
6832: THTTPContentDecoderContentType', '( crctFixedSize, crctChunked, crctUnsize )
6833: THTTPContentDecoderChunkState', '( crcsChunkHeader, crcsContent,
6834: + crcsContentCRLF, crcsTrailer, crcsFinished )
6835: THTTPContentDecoderLogEvent', 'Procedure ( const Sender : THTTPContentDecoder; const LogMsg : String)
6836: SIRegister_THTTPContentDecoder(CL);
6837: THTTPContentReaderMechanism', '( hcrnEvent, hcrnString, hcrnStream, hcrnFile )
6838: FindClass('TOBJECT'),'THTTPContentReader
6839: THTTPContentReaderProc', 'Procedure ( const Sender : THTTPContentReader)
6840: THTTPContentReaderLogEvent', 'Procedure(const Sender:THTTPContentReader;const LogMsg:String;const
LogLevel: Int;
6841: SIRegister_THTTPContentReader(CL);
6842: THTTPContentWriterMechanism', '(hctmEvent, hctmString, hctmStream, hctmFile )
6843: FindClass('TOBJECT'),'THTTPContentWriter
6844: THTTPContentWriterLogEvent', 'Procedure (const Sender : THTTPContentWriter;const LogMsg:AnsiString);
6845: SIRegister_THTTPContentWriter(CL);
6846: Procedure SelfTestcHTTUtils
6847: end;
6848:
6849: (*-----*)
6850: procedure SIRegister_cTLSUtils(CL: TPSPascalCompiler);
6851: begin
6852: 'TLSLibraryVersion','String').SetString( '1.00
6853: 'TLSError_None','LongInt').SetInt( 0);
6854: 'TLSError_InvalidBuffer','LongInt').SetInt( 1);
6855: 'TLSError_InvalidParameter','LongInt').SetInt( 2);
6856: 'TLSError_InvalidCertificate','LongInt').SetInt( 3);
6857: 'TLSError_InvalidState','LongInt').SetInt( 4);
6858: 'TLSError_DecodeError','LongInt').SetInt( 5);
6859: 'TLSError_BadProtocol','LongInt').SetInt( 6);
6860: Function TLSErrorMessage(const TLSError : Integer) : String
6861: SIRegister_ETLSError(CL);
6862: TTLSProtocolVersion', 'record major : Byte; minor : Byte; end
6863: PTLSPProtocolVersion', '^TTLSProtocolVersion // will not work
6864: Procedure InitSSLProtocolVersion30( var A : TTLSProtocolVersion)
6865: Procedure InitTLSProtocolVersion10( var A : TTLSProtocolVersion)
6866: Procedure InitTLSProtocolVersion11( var A : TTLSProtocolVersion)
6867: Procedure InitTLSProtocolVersion12( var A : TTLSProtocolVersion)
6868: Function IsTLSProtocolVersion(const A, B : TTLSProtocolVersion) : Boolean
6869: Function IsSSL2(const A : TTLSProtocolVersion) : Boolean
6870: Function IsSSL3(const A : TTLSProtocolVersion) : Boolean
6871: Function IsTLS10(const A : TTLSProtocolVersion) : Boolean
6872: Function IsTLS11(const A : TTLSProtocolVersion) : Boolean
6873: Function IsTLS12(const A : TTLSProtocolVersion) : Boolean
6874: Function IsTLS10OrLater(const A : TTLSProtocolVersion) : Boolean
6875: Function IsTLS11OrLater(const A : TTLSProtocolVersion) : Boolean
6876: Function IsTLS12OrLater(const A : TTLSProtocolVersion) : Boolean
6877: Function IsFutureTLSVersion(const A : TTLSProtocolVersion) : Boolean
6878: Function IsKnownTLSVersion(const A : TTLSProtocolVersion) : Boolean
6879: Function TLSProtocolVersionToStr(const A : TTLSProtocolVersion) : String
6880: Function TLSProtocolVersionName(const A : TTLSProtocolVersion) : String
6881: PTLSPRandom', '^TTLSRandom // will not work
6882: Procedure InitTLSPRandom( var Random : TTLSRandom)
6883: Function TLSRandomToStr(const Random : TTLSRandom) : AnsiString
6884: 'TLSSessionIDMaxLen','LongInt').SetInt( 32);
6885: Procedure InitTLSSessionID( var SessionID : TTLSSessionID; const A : AnsiString)
6886: Function EncodeTLSSessionID(var Buffer:string;const Size:Int;const SessionID:TTLSSessionID):Int;
6887: Function DecodeTLSSessionID(const Buffer:string;const Size:Int;var SessionID:TTLSSessionID):Int;
6888: TTLSSignatureAndHashAlgorithm', 'record Hash : TTLSHashAlgorithm
6889: +; Signature : TTLSSignatureAlgorithm; end
6890: // PTLSSignatureAndHashAlgorithm', '^TTLSSignatureAndHashAlgorithm +// will not work
6891: TTLSSignatureAndHashAlgorithmArray', 'array of TTLSSignatureAndHashAlgorithm
6892: TTLSKeyExchangeAlgorithm', '( tlskeaNone, tlskeaNULL, tlskeaDHE
6893: +DSS, tlskeaDHE_RSA, tlskeaDH_Anon, tlskeaRSA, tlskeaDH_DSS, tlskeaDH_RSA )
6894: TTLSMACAlgorithm', '( tlsmaNone, tlsmaNULL, tlsmaHMAC_MD5, tlsma
6895: +HMAC_SHA1, tlsmaHMAC_SHA256, tlsmaHMAC_SHA384, tlsmaHMAC_SHA512 )

```



```

6896:   TTLSMacAlgorithmInfo', 'record Name : AnsiString; DigestSize : I
6897:   +integer; Supported : Boolean; end
6898:   PTLSTMacAlgorithmInfo', '^TTLSMacAlgorithmInfo // will not work
6899:   'TLS_MAC_MAXDIGESTSIZE', 'LongInt').SetInt( 64);
6900:   TTLSPRFAlgorithm', '( tlpshaSHA256 )
6901:   Function tlp_MD5( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6902:   Function tlp_SHA1( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6903:   Function tlp_SHA256( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6904:   Function tlp_SHA512( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6905:   Function tlp10PRF( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6906:   Function tlp12PRF_SHA256( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6907:   Function tlp12PRF_SHA512( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6908:   Function TLSPRF(const ProtoVersion:TTLSProtocolVersion;const Secret,ALabel,Seed:AStrng;const
Size:Int):AStrng;
6909:   Function tlp10KeyBlock(const MasterSecret, ServerRandom,ClientRandom:AnsiString; const
Size:Integer):AnsiString
6910:   Function tlp12SHA256KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
Size:Int):AnsiString;
6911:   Function tlp12SHA512KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
Size:Int):AnsiString;
6912:   Function TLSKeyBlock(const ProtocolVersion:TTLSProtocolVersion; const MasterSecret, ServerRandom,
ClientRandom : AnsiString; const Size : Integer) : AnsiString
6913:   Function tlp10MasterSecret(const PreMasterSecret,ClientRandom, ServerRandom:AnsiString) :AnsiString;
6914:   Function tlp12SHA256MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString):AnsiString;
6915:   Function tlp12SHA512MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString): AnsiString;
6916:   Function TLSTMasterSecret( const ProtocolVersion: TTLSProtocolVersion;const PreMasterSecret,ClientRandom,
ServerRandom:AnsiString) : AnsiString
6917:   TTLSKeys', 'record KeyBlock : AnsiString; ClientMACKey : AnsiStr
6918:   +ing; ServerMACKey : AnsiString; ClientEncKey : AnsiString; ServerEncKey : '
6919:   +'AnsiString; ClientIV : AnsiString; ServerIV : AnsiString; end
6920:   Procedure GenerateTLSKeys( const ProtocolVersion : TTLSProtocolVersion; const MACKeyBits, CipherKeyBits,
IVBits : Integer; const MasterSecret, ServerRandom, ClientRandom : AnsiString; var TLSKeys : TTLSKeys)
6921:   Procedure GenerateFinalTLSKeys(const ProtocolVersion:TTLSProtocolVersion;const IsExportable:Boolean;const
ExpandedKeyBits:Integer;const ServerRandom,ClientRandom:AnsiString;var TLSKeys:TTLSKeys)
6922:   'TLS_PLAINTEXT_FRAGMENT_MAXSIZE', 'LongInt').SetInt( 16384 - 1);
6923:   'TLS_COMPRESSED_FRAGMENT_MAXSIZE', 'LongInt').SetInt( 16384 + 1024);
6924:   Procedure SelfTestcTLSUtils
6925:   end;
6926:
6927:   (*-----*)
6928:   procedure SIRegister_Reversi(CL: TPSPascalCompiler);
6929:   begin
6930:     sPosData', 'record corner : boolean; square2x2 : boolean; edge:boolean; stable : integer; internal :
integer; disks : integer; mx : integer; my : integer; end
6931:     // pBoard', '^tBoard // will not work
6932:     Function rCalculateData( cc : byte; cx, cy : integer) : sPosData
6933:     Function rCheckMove( color : byte; cx, cy : integer) : integer
6934:     //Function rDoStep( data : pBoard) : word
6935:     Function winExecAndWait( const sAppPath : string; wVisible : word) : boolean
6936:     end;
6937:
6938:   procedure SIRegister_IWDBCommon(CL: TPSPascalCompiler);
6939:   begin
6940:     Function InEditMode( ADataSet : TDataSet) : Boolean
6941:     Function CheckDataSource( ADataSource : TDataSource) : Boolean;
6942:     Function CheckDataSourceCel(ADataSource:TDataSource;const AFieldName:string;var VField:TField):boolean;
6943:     Function GetFieldText( AField : TField) : String
6944:     end;
6945:
6946:   procedure SIRegister_SortGrid(CL: TPSPascalCompiler);
6947:   begin
6948:     TPrintMode', '( pmPrint, pmPreview, pmPageCount )
6949:     TMyPrintRange', '( prAll, prSelected )
6950:     TSortStyle', '( ssAutomatic, ssNormal, ssNumeric, ssNumericExten'
6951:     +'ded, ssDateTime, ssTime, ssCustom )
6952:     TSortDirection', '( sdAscending, sdDescending )
6953:     TSetChecked', 'Procedure ( Sender : TObject; ACol, ARow : integer; State : Boolean)
6954:     TGetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integ'
6955:     +'er; var Strs : TStringList; var Width, Height : integer; var Sorted : Boolean)
6956:     TSetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integer; Str : String)
6957:     TSetEllipsis', 'Procedure ( Sender : TObject; ACol, ARow : integer)
6958:     SIRegister_TSortOptions(CL);
6959:     SIRegister_TPrintOptions(CL);
6960:     TSortedListEntry', 'record Str : String; RowNum : integer; SortOption : TSortOptions; end
6961:     SIRegister_TSortedList(CL);
6962:     TCellBevelStyle', '( cbNone, cbRaised, cbLowered )
6963:     TCellBevel', 'record Style: TCellBevelStyle; UpperLeftColor: TColor; LowerRightColor : TColor; end
6964:     TVertAlignment', '( taTopJustify, taBottomJustify, taMiddle )
6965:     TFormatOptions', 'record Brush : TBrush; Font : TFont; Alignment'
6966:     +'Horz : TAlignment; AlignmentVert : TVertAlignment; Bevel : TCellBevel; HideText : Boolean; end
6967:     SIRegister_TFontSetting(CL);
6968:     SIRegister_TFontList(CL);
6969:     AddTypeS(TFormatDrawCellEvent', 'Procedure ( Sender : TObject; Col, Row : '
6970:     +' integer;State:TGridDrawState;var FormatOptions:TFormatOptions;var CheckBox,Combobox,Ellipsis:Bool);
6971:     TSetFilterEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
6972:     TSearchEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
6973:     TUpdateGridEvent', 'Procedure ( Sender : TObject; ARow : integer)
6974:     TSizeChangedEvent', 'Procedure ( Sender : TObject; OldColCount, OldRowCount : integer)
6975:     TBeginSortEvent', 'Procedure ( Sender : TObject; var Col : integer)

```

```

6976: TEndSortEvent', 'Procedure ( Sender : TObject; Col : integer)
6977: TGetSortStyleEvent', 'Procedure ( Sender : TObject; Col : integer)
6978:   + 'r; var SortStyle : TSortStyle)
6979: TCellValidateEvent', 'Procedure ( Sender : TObject; ACol, ARow :
6980:   + ' integer; const OldValue : string; var NewValue : String; var Valid : Boolean)
6981: SIRegister_TSortGrid(CL);
6982: Function ExtendedCompare( const Str1, Str2 : String) : Integer
6983: Function NormalCompare( const Str1, Str2 : String) : Integer
6984: Function DateTimeCompare( const Str1, Str2 : String) : Integer
6985: Function NumericCompare( const Str1, Str2 : String) : Integer
6986: Function TimeCompare( const Str1, Str2 : String) : Integer
6987: //Function Compare( Item1, Item2 : Pointer) : Integer
6988: end;
6989:
6990: ***** procedure Register_IB(CL: TPSPascalCompiler);
6991: Procedure IBAlloc( var P, OldSize, NewSize : Integer)
6992: Procedure IBError( ErrMess : TIBClientError; const Args : array of const)
6993: Procedure IBDataBaseError
6994: Function StatusVector : PISC_STATUS
6995: Function StatusVectorArray : PStatusVector
6996: Function CheckStatusVector( ErrorCodes : array of ISC_STATUS) : Boolean
6997: Function StatusVectorAsText : string
6998: Procedure SetIBDataBaseErrorMessages( Value : TIBDataBaseErrorMessages)
6999: Function GetIBDataBaseErrorMessages : TIBDataBaseErrorMessages
7000:
7001:
7002: //*****unit uPSI_BoldUtils;*****
7003: Function CharCount( c : char; const s : string) : integer
7004: Function BoldNamesEqual( const name1, name2 : string) : Boolean
7005: Procedure BoldAppendToStrings(strings: TStrings; const aString: string; const ForceNewLine:Boolean)
7006: Function BoldSeparateStringList(strings:TStringList;const Separator,PreString,PostString:String):String
7007: Function BoldSeparatedAppend( const S1, S2 : string; const Separator : string) : string
7008: Function BoldTrim( const S : string) : string
7009: Function BoldIsPrefix( const S, Prefix : string) : Boolean
7010: Function BoldStrEqual( P1, P2 : PChar; Len : integer) : Boolean
7011: Function BoldStrAnsiEqual( P1, P2 : PChar; Len : integer) : Boolean
7012: Function BoldAnsiEqual( const S1, S2 : string) : Boolean
7013: Function BoldStrStringEqual( const S1 : string; P2 : PChar; Len : integer) : Boolean
7014: Function BoldCaseIndependentPos( const Substr, S : string) : Integer
7015: //Procedure EnumToStrings( aTypeInfo : pTypeInfo; Strings : TStrings)
7016: Function CapitalisedToSpaced( Capitalised : String) : String
7017: Function SpacedToCapitalised( Spaced : String) : String
7018: Function BooleanToString( BoolValue : Boolean) : String
7019: Function StringToBoolean( StrValue : String) : Boolean
7020: Function GetUpperLimitForMultiplicity( const Multiplicity : String) : Integer
7021: Function GetLowerLimitForMultiplicity( const Multiplicity : String) : Integer
7022: Function StringListToVarArray( List : TStringList) : variant
7023: Function IsLocalMachine( const Machinename : WideString) : Boolean
7024: Function GetComputerNameStr : string
7025: Function TimeStampComp( const Time1, Time2 : TTimeStamp) : Integer
7026: Function BoldStrToDateFmt(const S:string;const DateFormat:string;const DateSeparatorChar:char):TDateTime
7027: Function BoldDateToStrFmt(const aDate:TDateTime;DateFormat:string;const DateSeparatorChar:char):String;
7028: Function BoldParseFormattedDateList(const value:String;const formats:TStrings;var Date:TDateTime):Boolean;
7029: Function BoldParseFormattedDate(const value:String;const formats:array of string; var
Date:TDateTime):Boolean;
7030: Procedure EnsureTrailing( var Str : String; ch : char)
7031: Function BoldDirectoryExists( const Name : string) : Boolean
7032: Function BoldForceDirectories( Dir : string) : Boolean
7033: Function BoldRootRegistryKey : string
7034: Function GetModuleFileNameAsString( IncludePath : Boolean) : string
7035: Function BoldVariantToStrings( V : OleVariant; Strings : TStrings) : Integer
7036: Function LogicalAnd( A, B : Integer) : Boolean
7037: record TByHandleFileInformation dwFileAttributes : DWORD; '
7038:   + 'ftCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime '
7039:   + ' : TFileTime; dwVolumeSerialNumber : DWORD; nFileSizeHigh : DWORD; nFileSiz'
7040:   + 'eLow : DWORD; nNumberOfLinks : DWORD; nFileIndexHigh : DWORD; nFileIndexLow : DWORD; end
7041: Function GetFileInformationByHandle(hFile:THandle;var lpFileInformation:TByHandleFileInformation):BOOL;
7042: Function IsFirstInstance : Boolean
7043: Procedure ActivateFirst( AString : PChar)
7044: Procedure ActivateFirstCommandLine
7045: function MakeAckPkt(const BlockNumber: Word): string;
7046: procedure SendError(UDPBase:TiUDPBase;APeerIP:string;const APort:Integer;const ErrNumber:Word;ErrStr:string;
7047: procedure SendError(UDPClient: TiUDPClient; const ErrNumber: Word; ErrorString: string); overload;
7048: procedure SendError(UDPBase: TiUDPBase; APeerIP: string; const APort: Integer; E: Exception); overload;
7049: procedure SendError(UDPClient: TiUDPClient; E: Exception); overload;
7050: function IdStrToWord(const Value: String): Word;
7051: function IdWordToStr(const Value: Word): WordStr;
7052: Function HasInstructionSet( const InstructionSet : TCPUInstructionSet) : Boolean
7053: Function CPUFeatures : TCPUFeatures
7054:
7055: procedure SIRegister_xrttl_util_CPUUtils(CL: TPSPascalCompiler);
7056: begin
7057:   AddTypeS('TXRTLBitIndex', 'Integer
7058: Function XRTLSwapBits( Data : Cardinal; Bit1Index, Bit2Index : TXRTLBitIndex) : Cardinal
7059: Function XRTLBitTest( Data : Cardinal; BitIndex : TXRTLBitIndex) : Boolean
7060: Function XRTLBitSet( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7061: Function XRTLBitReset( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7062: Function XRTLBitComplement( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7063: Function XRTLSwapHiLo16( X : Word) : Word

```

```

7064: Function XRTLSwapHiLo32( X : Cardinal ) : Cardinal
7065: Function XRTLSwapHiLo64( X : Int64 ) : Int64
7066: Function XRTLROL32( A, S : Cardinal ) : Cardinal
7067: Function XRTLROL32( A, S : Cardinal ) : Cardinal
7068: Function XRTLROL16( A : Word; S : Cardinal ) : Word
7069: Function XRTLROL16( A : Word; S : Cardinal ) : Word
7070: Function XRTLROL8( A : Byte; S : Cardinal ) : Byte
7071: Function XRTLROL8( A : Byte; S : Cardinal ) : Byte
7072: //Procedure XRTLXorBlock( I1, I2, O1 : PByteArray; Len : integer)
7073: //Procedure XRTLIncBlock( P : PByteArray; Len : integer)
7074: Procedure XRTLUMul64( const A, B : Integer; var Mull, MulH : Integer)
7075: Function XRTLPopulation( A : Cardinal ) : Cardinal
7076: end;
7077:
7078: Function XRTLURLDecode( const ASrc : WideString ) : WideString
7079: Function XRTLURLEncode( const ASrc : WideString ) : WideString
7080: Function XRTLURINormalize( const AURI : WideString ) : WideString
7081: Procedure XRTLURIParse(const AURI:Widestring;var VProtocol,VHost,VPath,VDocument,VPort,VBookmark,VUserName,
VPassword : WideString)
7082: Function XRTLExtractLongPathName(APath: string): string;
7083:
7084: procedure SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
7085: begin
7086:   AddTypeS('Int8', 'ShortInt
7087:   AddTypeS('Int16', 'SmallInt
7088:   AddTypeS('Int32', 'LongInt
7089:   AddTypeS('UInt8', 'Byte
7090:   AddTypeS('UInt16', 'Word
7091:   AddTypeS('UInt32', 'LongWord
7092:   AddTypeS('UInt64', 'Int64
7093:   AddTypeS('Word8', 'UInt8
7094:   AddTypeS('Word16', 'UInt16
7095:   AddTypeS('Word32', 'UInt32
7096:   AddTypeS('Word64', 'UInt64
7097:   AddTypeS('LargeInt', 'Int64
7098:   AddTypeS('NativeInt', 'Integer
7099:   AddTypeS('NativeUInt', 'Cardinal
7100:   Const('BitsPerByte','LongInt').SetInt( 8);
7101:   Const('BitsPerWord','LongInt').SetInt( 16);
7102:   Const('BitsPerLongWord','LongInt').SetInt( 32);
7103:   //Const('BitsPerCardinal','LongInt').SetInt( BytesPerCardinal * 8);
7104:   //Const('BitsPerNativeWord','LongInt').SetInt( BytesPerNativeWord * 8);
7105:   Function MinI( const A, B : Integer ) : Integer
7106:   Function MaxI( const A, B : Integer ) : Integer
7107:   Function MinC( const A, B : Cardinal ) : Cardinal
7108:   Function MaxC( const A, B : Cardinal ) : Cardinal
7109:   Function SumClipI( const A, I : Integer ) : Integer
7110:   Function SumClipC( const A : Cardinal; const I : Integer ) : Cardinal
7111:   Function InByteRange( const A : Int64 ) : Boolean
7112:   Function InWordRange( const A : Int64 ) : Boolean
7113:   Function InLongWordRange( const A : Int64 ) : Boolean
7114:   Function InShortIntRange( const A : Int64 ) : Boolean
7115:   Function InSmallIntRange( const A : Int64 ) : Boolean
7116:   Function InLongIntRange( const A : Int64 ) : Boolean
7117:   AddTypeS('Bool8', 'ByteBool
7118:   AddTypeS('Bool16', 'WordBool
7119:   AddTypeS('Bool32', 'LongBool
7120:   AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )
7121:   AddTypeS('TCompareResultSet', 'set of TCompareResult
7122:   Function ReverseCompareResult( const C : TCompareResult ) : TCompareResult
7123:   Const('MinSingle','Single').setExtended( 1.5E-45);
7124:   Const('MaxSingle','Single').setExtended( 3.4E+38);
7125:   Const('MinDouble','Double').setExtended( 5.0E-324);
7126:   Const('MaxDouble','Double').setExtended( 1.7E+308);
7127:   Const('MinExtended','Extended').setExtended(3.4E-4932);
7128:   Const('MaxExtended','Extended').setExtended(1.1E+4932);
7129:   Const('MinCurrency','Currency').SetExtended( - 922337203685477.5807);
7130:   Const('MaxCurrency','Currency').SetExtended( 922337203685477.5807);
7131:   Function MinF( const A, B : Float ) : Float
7132:   Function MaxF( const A, B : Float ) : Float
7133:   Function ClipF( const Value : Float; const Low, High : Float ) : Float
7134:   Function InSingleRange( const A : Float ) : Boolean
7135:   Function InDoubleRange( const A : Float ) : Boolean
7136:   Function InCurrencyRange( const A : Float ) : Boolean;
7137:   Function InCurrencyRange1( const A : Int64 ) : Boolean;
7138:   Function FloatExponentBase2( const A : Extended; var Exponent : Integer ) : Boolean
7139:   Function FloatExponentBase10( const A : Extended; var Exponent : Integer ) : Boolean
7140:   Function FloatIsInfinity( const A : Extended ) : Boolean
7141:   Function FloatIsNaN( const A : Extended ) : Boolean
7142:   Const('SingleCompareDelta','Extended').setExtended( 1.0E-34);
7143:   Const('DoubleCompareDelta','Extended').setExtended( 1.0E-280);
7144:   Const('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
7145:   Const('DefaultCompareDelta','Extended').SetExtended( 1.0E-34);
7146:   Function FloatZero( const A : Float; const CompareDelta : Float ) : Boolean
7147:   Function FloatOne( const A : Float; const CompareDelta : Float ) : Boolean
7148:   Function FloatsEqual( const A, B : Float; const CompareDelta : Float ) : Boolean
7149:   Function FloatsCompare( const A, B : Float; const CompareDelta : Float ) : TCompareResult
7150:   Const('SingleCompareEpsilon','Extended').setExtended( 1.0E-5);
7151:   Const('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13);

```

```

7152: Const('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17);
7153: Const('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
7154: Function ApproxEqual( const A, B : Extended; const CompareEpsilon : Double) : Boolean
7155: Function ApproxCompare( const A, B : Extended; const CompareEpsilon : Double): TCompareResult
7156: Function cClearBit( const Value, BitIndex : LongWord) : LongWord
7157: Function cSetBit( const Value, BitIndex : LongWord) : LongWord
7158: Function cIsBitSet( const Value, BitIndex : LongWord) : Boolean
7159: Function cToggleBit( const Value, BitIndex : LongWord) : LongWord
7160: Function cIsHighBitSet( const Value : LongWord) : Boolean
7161: Function SetBitScanForward( const Value : LongWord) : Integer;
7162: Function SetBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7163: Function SetBitScanReverse( const Value : LongWord) : Integer;
7164: Function SetBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7165: Function ClearBitScanForward( const Value : LongWord) : Integer;
7166: Function ClearBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7167: Function ClearBitScanReverse( const Value : LongWord) : Integer;
7168: Function ClearBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7169: Function cReverseBits( const Value : LongWord) : LongWord;
7170: Function cReverseBits1( const Value : LongWord; const BitCount : Integer) : LongWord;
7171: Function cSwapEndian( const Value : LongWord) : LongWord
7172: Function cTwosComplement( const Value : LongWord) : LongWord
7173: Function RotateLeftBits16( const Value : Word; const Bits : Byte) : Word
7174: Function RotateLeftBits32( const Value : LongWord; const Bits : Byte) : LongWord
7175: Function RotateRightBits16( const Value : Word; const Bits : Byte) : Word
7176: Function RotateRightBits32( const Value : LongWord; const Bits : Byte) : LongWord
7177: Function cBitCount( const Value : LongWord) : LongWord
7178: Function cIsPowerOfTwo( const Value : LongWord) : Boolean
7179: Function LowBitMask( const HighBitIndex : LongWord) : LongWord
7180: Function HighBitMask( const LowBitIndex : LongWord) : LongWord
7181: Function RangeBitMask( const LowBitIndex, HighBitIndex : LongWord) : LongWord
7182: Function SetBitRange( const Value : LongWord; const LowBitIndex, HighBitIndex : LongWord) : LongWord
7183: Function ClearBitRange( const Value : LongWord; const LowBitIndex, HighBitIndex : LongWord) : LongWord
7184: Function ToggleBitRange( const Value : LongWord; const LowBitIndex, HighBitIndex : LongWord) : LongWord
7185: Function IsBitRangeSet( const Value : LongWord; const LowBitIndex, HighBitIndex : LongWord) : Boolean
7186: Function IsBitRangeClear( const Value : LongWord; const LowBitIndex, HighBitIndex : LongWord) : Boolean
7187: // AddTypes('CharSet', 'set of AnsiChar
7188: AddTypes('CharSet', 'set of Char' //!!!
7189: AddTypes('AnsiCharSet', 'TCharSet
7190: AddTypes('ByteSet', 'set of Byte
7191: AddTypes('AnsiChar', 'Char
7192: // Function AsCharSet( const C : array of AnsiChar) : CharSet
7193: Function AsByteSet( const C : array of Byte) : ByteSet
7194: Procedure ComplementChar( var C : CharSet; const Ch : Char)
7195: Procedure ClearCharSet( var C : CharSet)
7196: Procedure FillCharSet( var C : CharSet)
7197: Procedure ComplementCharSet( var C : CharSet)
7198: Procedure AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7199: Procedure Union( var DestSet : CharSet; const SourceSet : CharSet)
7200: Procedure Difference( var DestSet : CharSet; const SourceSet : CharSet)
7201: Procedure Intersection( var DestSet : CharSet; const SourceSet : CharSet)
7202: Procedure XORCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7203: Function IsSubSet( const A, B : CharSet) : Boolean
7204: Function IsEqual( const A, B : CharSet) : Boolean
7205: Function IsEmpty( const C : CharSet) : Boolean
7206: Function IsComplete( const C : CharSet) : Boolean
7207: Function cCharCount( const C : CharSet) : Integer
7208: Procedure ConvertCaseInsensitive( var C : CharSet)
7209: Function CaseInsensitiveCharSet( const C : CharSet) : CharSet
7210: Function IntRangeLength( const Low, High : Integer) : Int64
7211: Function IntRangeAdjacent( const Low1, High1, Low2, High2 : Integer) : Boolean
7212: Function IntRangeOverlap( const Low1, High1, Low2, High2 : Integer) : Boolean
7213: Function IntRangeHasElement( const Low, High, Element : Integer) : Boolean
7214: Function IntRangeIncludeElement( var Low, High : Integer; const Element : Integer) : Boolean
7215: Function IntRangeIncludeElementRange( var Low, High : Integer; const LowElement, HighElement : Integer) : Boolean
7216: Function CardRangeLength( const Low, High : Cardinal) : Int64
7217: Function CardRangeAdjacent( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7218: Function CardRangeOverlap( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7219: Function CardRangeHasElement( const Low, High, Element : Cardinal) : Boolean
7220: Function CardRangeIncludeElement( var Low, High : Cardinal; const Element : Cardinal) : Boolean
7221: Function CardRangeIncludeElementRange( var Low, High : Cardinal; const LowElement, HighElement : Cardinal) : Boolean;
7222: AddTypes('UnicodeChar', 'WideChar
7223: Function Compare( const I1, I2 : Boolean) : TCompareResult;
7224: Function Compare1( const I1, I2 : Integer) : TCompareResult;
7225: Function Compare2( const I1, I2 : Int64) : TCompareResult;
7226: Function Compare3( const I1, I2 : Extended) : TCompareResult;
7227: Function CompareA( const I1, I2 : AnsiString) : TCompareResult
7228: Function CompareW( const I1, I2 : WideString) : TCompareResult
7229: Function cSgn( const A : LongInt) : Integer;
7230: Function cSgn1( const A : Int64) : Integer;
7231: Function cSgn2( const A : Extended) : Integer;
7232: AddTypes('TConvertResult', '( convertOK, convertFormatError, convertOverflow )
7233: Function AnsiCharToInt( const A : AnsiChar) : Integer
7234: Function WideCharToInt( const A : WideChar) : Integer
7235: Function CharToInt( const A : Char) : Integer
7236: Function IntToAnsiChar( const A : Integer) : AnsiChar
7237: Function IntToWideChar( const A : Integer) : WideChar
7238: Function IntToChar( const A : Integer) : Char
7239: Function IsHexAnsiChar( const Ch : AnsiChar) : Boolean
7240: Function IsHexWideChar( const Ch : WideChar) : Boolean

```



```

7241: Function IsHexChar( const Ch : Char) : Boolean
7242: Function HexAnsiCharToInt( const A : AnsiChar) : Integer
7243: Function HexWideCharToInt( const A : WideChar) : Integer
7244: Function HexCharToInt( const A : Char) : Integer
7245: Function IntToUpperHexAnsiChar( const A : Integer) : AnsiChar
7246: Function IntToUpperHexWideChar( const A : Integer) : WideChar
7247: Function IntToUpperHexChar( const A : Integer) : Char
7248: Function IntToLowerHexAnsiChar( const A : Integer) : AnsiChar
7249: Function IntToLowerHexWideChar( const A : Integer) : WideChar
7250: Function IntToLowerHexChar( const A : Integer) : Char
7251: Function IntToStringA( const A : Int64) : AnsiString
7252: Function IntToStringW( const A : Int64) : WideString
7253: Function IntToString( const A : Int64) : String
7254: Function UIntToStringA( const A : NativeUInt) : AnsiString
7255: Function UIntToStringW( const A : NativeUInt) : WideString
7256: Function UIntToString( const A : NativeUInt) : String
7257: Function LongWordToStrA( const A : LongWord; const Digits : Integer) : AnsiString
7258: Function LongWordToStrW( const A : LongWord; const Digits : Integer) : WideString
7259: Function LongWordToStrU( const A : LongWord; const Digits : Integer) : UnicodeString
7260: Function LongWordToStr( const A : LongWord; const Digits : Integer) : String
7261: Function LongWordToHexA( const A:LongWord;const Digits:Integer;const UpperCase:Boolean):AnsiString;
7262: Function LongWordToHexW( const A:LongWord;const Digits:Integer;const UpperCase:Boolean):WideString;
7263: Function LongWordToHex( const A : LongWord; const Digits : Integer;const UpperCase:Boolean):String
7264: Function LongWordToOctA( const A : LongWord; const Digits : Integer) : AnsiString
7265: Function LongWordToOctW( const A : LongWord; const Digits : Integer) : WideString
7266: Function LongWordToOct( const A : LongWord; const Digits : Integer) : String
7267: Function LongWordToBinA( const A : LongWord; const Digits : Integer) : AnsiString
7268: Function LongWordToBinW( const A : LongWord; const Digits : Integer) : WideString
7269: Function LongWordToBin( const A : LongWord; const Digits : Integer) : String
7270: Function TryStringToInt64A( const S : AnsiString; out A : Int64) : Boolean
7271: Function TryStringToInt64W( const S : WideString; out A : Int64) : Boolean
7272: Function TryStringToInt64( const S : String; out A : Int64) : Boolean
7273: Function StringToInt64DefA( const S : AnsiString; const Default : Int64) : Int64
7274: Function StringToInt64DefW( const S : WideString; const Default : Int64) : Int64
7275: Function StringToInt64Def( const S : String; const Default : Int64) : Int64
7276: Function StringToInt64A( const S : AnsiString) : Int64
7277: Function StringToInt64W( const S : WideString) : Int64
7278: Function StringToInt64( const S : String) : Int64
7279: Function TryStringToIntA( const S : AnsiString; out A : Integer) : Boolean
7280: Function TryStringToIntW( const S : WideString; out A : Integer) : Boolean
7281: Function TryStringToInt( const S : String; out A : Integer) : Boolean
7282: Function StringToIntDefA( const S : AnsiString; const Default : Integer) : Integer
7283: Function StringToIntDefW( const S : WideString; const Default : Integer) : Integer
7284: Function StringToIntDef( const S : String; const Default : Integer) : Integer
7285: Function StringToIntA( const S : AnsiString) : Integer
7286: Function StringToIntW( const S : WideString) : Integer
7287: Function StringToInt( const S : String) : Integer
7288: Function TryStringToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7289: Function TryStringToLongWordW( const S : WideString; out A : LongWord) : Boolean
7290: Function TryStringToLongWord( const S : String; out A : LongWord) : Boolean
7291: Function StringToLongWordA( const S : AnsiString) : LongWord
7292: Function StringToLongWordW( const S : WideString) : LongWord
7293: Function StringToLongWord( const S : String) : LongWord
7294: Function HexToUIntA( const S : AnsiString) : NativeUInt
7295: Function HexToUIntW( const S : WideString) : NativeUInt
7296: Function HexToUInt( const S : String) : NativeUInt
7297: Function TryHexToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7298: Function TryHexToLongWordW( const S : WideString; out A : LongWord) : Boolean
7299: Function TryHexToLongWord( const S : String; out A : LongWord) : Boolean
7300: Function HexToLongWordA( const S : AnsiString) : LongWord
7301: Function HexToLongWordW( const S : WideString) : LongWord
7302: Function HexToLongWord( const S : String) : LongWord
7303: Function TryOctToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7304: Function TryOctToLongWordW( const S : WideString; out A : LongWord) : Boolean
7305: Function TryOctToLongWord( const S : String; out A : LongWord) : Boolean
7306: Function OctToLongWordA( const S : AnsiString) : LongWord
7307: Function OctToLongWordW( const S : WideString) : LongWord
7308: Function OctToLongWord( const S : String) : LongWord
7309: Function TryBinToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7310: Function TryBinToLongWordW( const S : WideString; out A : LongWord) : Boolean
7311: Function TryBinToLongWord( const S : String; out A : LongWord) : Boolean
7312: Function BinToLongWordA( const S : AnsiString) : LongWord
7313: Function BinToLongWordW( const S : WideString) : LongWord
7314: Function BinToLongWord( const S : String) : LongWord
7315: Function FloatToStringA( const A : Extended) : AnsiString
7316: Function FloatToStringW( const A : Extended) : WideString
7317: Function FloatToString( const A : Extended) : String
7318: Function TryStringToFloatA( const A : AnsiString; out B : Extended) : Boolean
7319: Function TryStringToFloatW( const A : WideString; out B : Extended) : Boolean
7320: Function TryStringToFloat( const A : String; out B : Extended) : Boolean
7321: Function StringToFloatA( const A : AnsiString) : Extended
7322: Function StringToFloatW( const A : WideString) : Extended
7323: Function StringToFloat( const A : String) : Extended
7324: Function StringToFloatDefA( const A : AnsiString; const Default : Extended) : Extended
7325: Function StringToFloatDefW( const A : WideString; const Default : Extended) : Extended
7326: Function StringToFloatDef( const A : String; const Default : Extended) : Extended
7327: Function EncodeBase64(const S,Alphabet:AnsiString; const Pad:Boolean; const PadMultiple:Integer; const PadChar: AnsiChar) : AnsiString
7328: Function DecodeBase64( const S, Alphabet : AnsiString; const PadSet : CharSet) : AnsiString

```

```

7329: unit uPSI_cFundamentUtils;
7330: Const('b64_MIMEBase64','Str').String('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
7331: Const('b64_UUEncode','String').String('!#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNQRSTUUVWXYZ[\]^_';
7332: Const('b64_XXEncode','String').String('+-0123456789ABCDEFGHIJKLMNQRSTUUVWXYZabcdefghijklmnopqrstuvwxyz';
7333: Const('CCHARSET','String').SetString(b64_XXEncode);
7334: Const('CHEXSET','String').SetString('0123456789ABCDEF
7335: Const('HEXDIGITS','String').SetString('0123456789ABCDEF
7336: StHexDigits : array[0..$F] of Char = '0123456789ABCDEF';
7337: Const('DIGISET','String').SetString('0123456789
7338: Const('LETTERSET','String').SetString('ABCDEFGHIJKLMNQRSTUUVWXYZ'
7339: Const('DIGISET2','TCharset').SetSet('0123456789'
7340: Const('LETTERSET2','TCharset').SetSet('ABCDEFGHIJKLMNQRSTUUVWXYZ'
7341: Const('HEXSET2','TCharset').SetSet('0123456789ABCDEF');
7342: Const('NUMBERSET','TCharset').SetSet('0123456789');
7343: Const('NUMBERS','String').SetString('0123456789');
7344: Const('LETTERS','String').SetString('ABCDEFGHIJKLMNQRSTUUVWXYZ');
7345: Function CharSetToStr( const C : CharSet) : AnsiString
7346: Function StrToCharSet( const S : AnsiString) : CharSet
7347: Function MIMEBase64Decode( const S : AnsiString) : AnsiString
7348: Function MIMEBase64Encode( const S : AnsiString) : AnsiString
7349: Function UUDecode( const S : AnsiString) : AnsiString
7350: Function XXDecode( const S : AnsiString) : AnsiString
7351: Function BytesToHex( const P : array of Byte; const UpperCase : Boolean) : AnsiString
7352: Function InterfaceToStrA( const I : IInterface) : AnsiString
7353: Function InterfaceToStrW( const I : IInterface) : WideString
7354: Function InterfaceToStr( const I : IInterface) : String
7355: Function ObjectClassName( const O : TObject) : String
7356: Function ClassClassName( const C : TClass) : String
7357: Function ObjectToStr( const O : TObject) : String
7358: Function ObjectToString( const O : TObject) : String
7359: Function CharSetToStr( const C : CharSet) : AnsiString
7360: Function StrToCharSet( const S : AnsiString) : CharSet
7361: Function HashStrA(const S : AnsiString; const Index : Integer; const Count: Integer;const
AsciiCaseSensitive : Boolean; const Slots : LongWord) : LongWord
7362: Function HashStrW(const S:Widestring;const Index:Integer;const Count:Integer;const
AsciiCaseSensitive:Boolean; const Slots:LongWord) : LongWord
7363: Function HashStr(const S : String; const Index : Integer; const Count : Integer; const AsciiCaseSensitive
: Boolean; const Slots : LongWord) : LongWord
7364: Function HashInteger( const I : Integer; const Slots : LongWord) : LongWord
7365: Function HashLongWord( const I : LongWord; const Slots : LongWord) : LongWord
7366: Const('Bytes1KB','LongInt').SetInt( 1024);
7367: SIRegister_IInterface(CL);
7368: Procedure SelfTestCFundamentUtils
7369:
7370: Function CreateSchedule : IJclSchedule
7371: Function NullStamp : TTimeStamp
7372: Function CompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
7373: Function EqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
7374: Function IsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
7375:
7376: procedure SIRegister_uwinplot(CL: TPSPascalCompiler);
7377: begin
7378: AddTypeS('TFunc', 'function(X : Float) : Float;
7379: Function InitGraphics( Width, Height : Integer) : Boolean
7380: Procedure SetWindow( Canvas : TCanvas; X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
7381: Procedure SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
7382: Procedure SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
7383: Procedure GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float)
7384: Procedure GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float)
7385: Procedure SetGraphTitle( Title : String)
7386: Procedure SetOxTitle( Title : String)
7387: Procedure SetOyTitle( Title : String)
7388: Function GetGraphTitle : String
7389: Function GetOxTitle : String
7390: Function GetOyTitle : String
7391: Procedure PlotOxAxis( Canvas : TCanvas)
7392: Procedure PlotOyAxis( Canvas : TCanvas)
7393: Procedure PlotGrid( Canvas : TCanvas; Grid : TGrid)
7394: Procedure WriteGraphTitle( Canvas : TCanvas)
7395: Function SetMaxCurv( NCurv : Byte) : Boolean
7396: Procedure SetPointParam( CurvIndex, Symbol, Size : Integer; Color : TColor)
7397: Procedure SetLineParam( CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor)
7398: Procedure SetCurvLegend( CurvIndex : Integer; Legend : String)
7399: Procedure SetCurvStep( CurvIndex, Step : Integer)
7400: Function GetMaxCurv : Byte
7401: Procedure GetPointParam( CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor)
7402: Procedure GetLineParam(CurvIndex:Integer;var Style:TPenStyle;var Width:Integer; var Color:TColor);
7403: Function GetCurvLegend( CurvIndex : Integer) : String
7404: Function GetCurvStep( CurvIndex : Integer) : Integer
7405: Procedure PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Integer)
7406: Procedure PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Integer)
7407: Procedure PlotCurveWithErrorBars(Canvas : TCanvas; X,Y,S: TVector; Ns,Lb,Ub,CurvIndex:Integer)
7408: Procedure PlotFunc(Canvas: TCanvas; Func: TFunc; Xmin,Xmax: Float; Npt,CurvIndex: Integer)
7409: Procedure WriteLegend( Canvas : TCanvas; NCurv : Integer; ShowPoints, ShowLines : Boolean)
7410: Procedure ConRec( Cnvs : TCanvas; Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
7411: Function Xpixel( X : Float) : Integer
7412: Function Ypixel( Y : Float) : Integer
7413: Function Xuser( X : Integer) : Float
7414: Function Yuser( Y : Integer) : Float

```

```

7415: end;
7416:
7417: Procedure FFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7418: Procedure IFFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7419: Procedure FFT_Integer( NumSamples : Integer; RealIn, ImagIn : TIntVector; OutArray : TCompVector)
7420: Procedure FFT_Integer_Cleanup
7421: Procedure CalcFrequency(NumSamples,FrequencyIndex: Integer;InArray: TCompVector;var FT : Complex)
7422: //unit uPSI_JclStreams;
7423: Function StreamSeek( Stream : TStream; const Offset : Int64; const Origin : TSeekOrigin) : Int64
7424: Function StreamCopy( Source : TStream; Dest : TStream; BufferSize : Integer) : Int64
7425: Function CompareStreams( A, B : TStream; BufferSize : Integer) : Boolean
7426: Function JCompareFiles( const FileA, FileB : TFileName; BufferSize : Integer) : Boolean
7427:
7428: procedure SIRegister_FmxUtils(CL: TPSPascalCompiler);
7429: begin
7430:   FindClass('TOBJECT'),'EInvalidDest
7431:   FindClass('TOBJECT'),'EFCantMove
7432:   Procedure fmxCopyFile( const FileName, DestName : string)
7433:   Procedure fmxMoveFile( const FileName, DestName : string)
7434:   Function fmxGetFileSize( const FileName : string) : LongInt
7435:   Function fmxFileDateTime( const FileName : string) : TDateTime
7436:   Function fmxHasAttr( const FileName : string; Attr : Word) : Boolean
7437:   Function fmxExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer):THandle;
7438: end;
7439:
7440: procedure SIRegister_FindFileIter(CL: TPSPascalCompiler);
7441: begin
7442:   SIRegister_IFindFileIterator(CL);
7443:   Function CreateFindFile(const Path:string; IncludeAttr:Integer;out iffi:IFindFileIterator):Bool;
7444: end;
7445:
7446: procedure SIRegister_PCharUtils(CL: TPSPascalCompiler);
7447: begin
7448:   Function SkipWhite( cp : PChar) : PChar
7449:   Function ReadStringDoubleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7450:   Function ReadStringSingleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7451:   Function ReadIdent( cp : PChar; var ident : string) : PChar
7452: end;
7453:
7454: procedure SIRegister_JclStrHashMap(CL: TPSPascalCompiler);
7455: begin
7456:   SIRegister_TStringHashMapTraits(CL);
7457:   Function CaseSensitiveTraits : TStringHashMapTraits
7458:   Function CaseInsensitiveTraits : TStringHashMapTraits
7459:   THashNode', 'record Str : string; Ptr : Pointer; Left : PHashNod'
7460:   + 'e; Right : PHashNode; end
7461:   //PHashArray', 'THashArray // will not work
7462:   SIRegister_TStringHashMap(CL);
7463:   THashValue', 'Cardinal
7464:   Function StrHash( const s : string) : THashValue
7465:   Function TextHash( const s : string) : THashValue
7466:   Function DataHash( var AValue, ASize : Cardinal) : THashValue
7467:   Function Iterate_FreeObjects( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7468:   Function Iterate_Dispose( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7469:   Function Iterate_FreeMem( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7470:   SIRegister_TCaseSensitiveTraits(CL);
7471:   SIRegister_TCaseInsensitiveTraits(CL);
7472:
7473:
7474: //*****unit uPSI_umath;
7475: Function uExpo( X : Float) : Float
7476: Function uExp2( X : Float) : Float
7477: Function uExp10( X : Float) : Float
7478: Function uLog( X : Float) : Float
7479: Function uLog2( X : Float) : Float
7480: Function uLog10( X : Float) : Float
7481: Function uLogA( X, A : Float) : Float
7482: Function uIntPower( X : Float; N : Integer): Float
7483: Function uPower( X, Y : Float) : Float
7484: Function SgnGamma( X : Float) : Integer
7485: Function Stirling( X : Float) : Float
7486: Function StirLog( X : Float) : Float
7487: Function Gamma( X : Float) : Float
7488: Function LnGamma( X : Float) : Float
7489: Function DiGamma( X : Float) : Float
7490: Function TriGamma( X : Float) : Float
7491: Function IGamma( X : Float) : Float
7492: Function JGamma( X : Float) : Float
7493: Function InvGamma( X : Float) : Float
7494: Function Erf( X : Float) : Float
7495: Function Erfc( X : Float) : Float
7496: Function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7497: { Correlation coefficient between samples X and Y }
7498: function DBeta(A, B, X : Float) : Float;
7499: { Density of Beta distribution with parameters A and B }
7500: Function LambertW( X : Float; UBranch, Offset : Boolean) : Float
7501: Function Beta(X, Y : Float) : Float
7502: Function Binomial( N, K : Integer) : Float
7503: Function PBinom( N : Integer; P : Float; K : Integer) : Float

```

```

7504: Procedure Cholesky( A, L : TMatrix; Lb, Ub : Integer)
7505: Procedure LU_Decom( A : TMatrix; Lb, Ub : Integer)
7506: Procedure LU_Solve( A : TMatrix; B : TVector; Lb, Ub : Integer; X : TVector)
7507: Function DNorm( X : Float) : Float
7508:
7509: function DGamma(A, B, X : Float) : Float;
7510: { Density of Gamma distribution with parameters A and B }
7511: function DKhi2(Nu : Integer; X : Float) : Float;
7512: { Density of Khi-2 distribution with Nu d.o.f. }
7513: function DStudent(Nu : Integer; X : Float) : Float;
7514: { Density of Student distribution with Nu d.o.f. }
7515: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float;
7516: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
7517: function IBeta(A, B, X : Float) : Float;
7518: { Incomplete Beta function}
7519: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7520:
7521: procedure SIRegister_unlfit(CL: TPSPascalCompiler);
7522: begin
7523:   Procedure SetOptAlgo( Algo : TOptAlgo)
7524: procedure SetOptAlgo(Algo : TOptAlgo);
7525: { -----
7526:   Sets the optimization algorithm according to Algo, which must be
7527:   NL_MARQ, NL_SIMP, NL_BFGS, NL_SA, NL_GA. Default is NL_MARQ   }
7528:
7529: Function GetOptAlgo : TOptAlgo
7530: Procedure SetMaxParam( N : Byte)
7531: Function GetMaxParam : Byte
7532: Procedure SetParamBounds( I : Byte; ParamMin, ParamMax : Float)
7533: Procedure GetParamBounds( I : Byte; var ParamMin, ParamMax : Float)
7534: Function NullParam( B : TVector; Lb, Ub : Integer) : Boolean
7535: Procedure NLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y : TVector; Lb, Ub : Integer; MaxIter :
Integer; Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7536: Procedure WNLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y, S : TVector; Lb, Ub:Integer;
MaxIter:Integer;Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7537: Procedure SetMCFile( FileName : String)
7538: Procedure SimFit(RegFunc:TRegFunc;X,Y:TVector;Lb,Ub:Integer;B:TVector;FirstPar,LastPar:Integer;V:TMatrix;
7539: Procedure WSimFit(RegFunc:TRegFunc; X,Y,S:TVector;Lb,Ub:Integer;B:TVector;FirstPar,
LastPar:Integer;V:TMatrix);
7540: end;
7541:
7542: (*-----*)
7543: procedure SIRegister_usimplex(CL: TPSPascalCompiler);
7544: begin
7545:   Procedure SaveSimplex( FileName : string)
7546:   Procedure Simplex(Func:TFuncNVar; X:TVector;Lb,Ub:Integer; MaxIter:Integer;Tol:Float; var F_min:Float);
7547: end;
7548: (*-----*)
7549: procedure SIRegister_uregtest(CL: TPSPascalCompiler);
7550: begin
7551:   Procedure RegTest(Y,Ycalc: TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest)
7552:   Procedure WRegTest(Y,Ycalc,S:TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest);
7553: end;
7554:
7555: procedure SIRegister_ustrings(CL: TPSPascalCompiler);
7556: begin
7557:   Function LTrim( S : String) : String
7558:   Function RTrim( S : String) : String
7559:   Function uTrim( S : String) : String
7560:   Function StrChar( N : Byte; C : Char) : String
7561:   Function RFill( S : String; L : Byte) : String
7562:   Function LFill( S : String; L : Byte) : String
7563:   Function CFill( S : String; L : Byte) : String
7564:   Function Replace( S : String; C1, C2 : Char) : String
7565:   Function Extract( S : String; var Index : Byte; Delim : Char) : String
7566:   Procedure Parse( S : String; Delim : Char; Field : TStrVector; var N : Byte)
7567:   Procedure SetFormat( NumLength, MaxDec : Integer; FloatPoint, NSZero : Boolean)
7568:   Function FloatStr( X : Float) : String
7569:   Function IntStr( N : LongInt) : String
7570:   Function uCompStr( Z : Complex) : String
7571: end;
7572:
7573: procedure SIRegister_uhyper(CL: TPSPascalCompiler);
7574: begin
7575:   Function uSinh( X : Float) : Float
7576:   Function uCosh( X : Float) : Float
7577:   Function uTanh( X : Float) : Float
7578:   Function uArcSinh( X : Float) : Float
7579:   Function uArcCosh( X : Float) : Float
7580:   Function ArcTanh( X : Float) : Float
7581:   Procedure SinhCosh( X : Float; var SinhX, CoshX : Float)
7582: end;
7583:
7584: procedure SIRegister_urandom(CL: TPSPascalCompiler);
7585: begin
7586: type RNG_Type =
7587:   (RNG_MWC,      { Multiply-With-Carry }
7588:   RNG_MT,        { Mersenne Twister }
7589:   RNG_UVAG);    { Universal Virtual Array Generator }

```



```

7590: Procedure SetRNG( RNG : RNG_Type)
7591: Procedure InitGen( Seed : RNG_IntType)
7592: Procedure SRand( Seed : RNG_IntType)
7593: Function IRanGen : RNG_IntType
7594: Function IRanGen31 : RNG_IntType
7595: Function RanGen1 : Float
7596: Function RanGen2 : Float
7597: Function RanGen3 : Float
7598: Function RanGen53 : Float
7599: end;
7600:
7601: // Optimization by Simulated Annealing
7602: procedure SIRegister_usimann(CL: TPSPascalCompiler);
7603: begin
7604:   Procedure InitSAParams( NT, NS, NCycles : Integer; RT : Float)
7605:   Procedure SA_CreateLogFile( FileName : String)
7606:   Procedure SimAnn(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7607: end;
7608:
7609: procedure SIRegister_uranuvag(CL: TPSPascalCompiler);
7610: begin
7611:   Procedure InitUVAGbyString( KeyPhrase : string)
7612:   Procedure InitUVAG( Seed : RNG_IntType)
7613:   Function IRanUVAG : RNG_IntType
7614: end;
7615:
7616: procedure SIRegister_uenalg(CL: TPSPascalCompiler);
7617: begin
7618:   Procedure InitGAParams( NP, NG : Integer; SR, MR, HR : Float)
7619:   Procedure GA_CreateLogFile( LogFileName : String)
7620:   Procedure GenAlg(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7621: end;
7622:
7623: TVector', 'array of Float
7624: procedure SIRegister_uqsort(CL: TPSPascalCompiler);
7625: begin
7626:   Procedure QSort( X : TVector; Lb, Ub : Integer)
7627:   Procedure DQSort( X : TVector; Lb, Ub : Integer)
7628: end;
7629:
7630: procedure SIRegister_uinterv(CL: TPSPascalCompiler);
7631: begin
7632:   Procedure Interval( X1, X2 : Float; MinDiv, MaxDiv : Integer; var Min, Max, Step : Float)
7633:   Procedure AutoScale(X: TVector; Lb, Ub : Integer; Scale : TScale; var XMin, XMax, XStep:Float)
7634: end;
7635:
7636: procedure SIRegister_D2XXUnit(CL: TPSPascalCompiler);
7637: begin
7638:   FT_Result', 'Integer
7639:   //TDWordptr', '^DWord // will not work
7640:   TFT_Program_Data', 'record Signature1 : DWord; Signature2 : DWord'
7641:   d; Version : DWord; VendorID : Word; ProductID : Word; Manufacturer : PChar
7642:   r; ManufacturerID : PChar; Description : PChar; SerialNumber : PChar; MaxP
7643:   ower : Word; PnP : Word; SelfPowered : Word; RemoteWakeUp : Word; Rev4 : B
7644:   yte; IsoIn : Byte; IsoOut : Byte; PullDownEnable : Byte; SerNumEnable : By
7645:   te; USBVersionEnable : Byte; USBVersion : Word; Rev5 : Byte; IsoInA : Byte
7646:   ; IsoInB : Byte; IsoOutA : Byte; IsoOutB : Byte; PullDownEnable5 : Byte; S
7647:   erNumEnable5 : Byte; USBVersionEnable5 : Byte; USBVersion5 : Word; AIsHigh
7648:   Current : Byte; BIsHighCurrent : Byte; IFAIsFifo : Byte; IFAIsFifoTar : By
7649:   te; IFAIsFastSer : Byte; AIsVCP : Byte; IFBIsFifo : Byte; IFBIsFifoTar : B
7650:   yte; IFBIsFastSer : Byte; BIsVCP : Byte; UseExtOsc : Byte; HighDriveIOs :
7651:   Byte; EndpointSize : Byte; PullDownEnableR : Byte; SerNumEnableR : Byte; I
7652:   nvertTXD : Byte; InvertRXD : Byte; InvertRTS : Byte; InvertCTS : Byte; Inv
7653:   ertDTR : Byte; InvertDSR : Byte; InvertDCD : Byte; InvertRI : Byte; Cbus0
7654:   : Byte; Cbus1 : Byte; Cbus2 : Byte; Cbus3 : Byte; Cbus4 : Byte; RIsVCP : B
7655:   yte; end
7656: end;
7657:
7658:
7659: //***** PaintFX*****
7660: procedure SIRegister_TJvPaintFX(CL: TPSPascalCompiler);
7661: begin
7662:   //with RegClassS(CL,'TComponent', 'TJvPaintFX') do
7663:   with AddClassN(FindClass('TComponent'),'TJvPaintFX') do begin
7664:     Procedure Solarize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7665:     Procedure Posterize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7666:     Procedure Blend( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7667:     Procedure Blend2( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7668:     Procedure ExtractColor( const Dst : TBitmap; AColor : TColor)
7669:     Procedure ExcludeColor( const Dst : TBitmap; AColor : TColor)
7670:     Procedure Turn( Src, Dst : TBitmap)
7671:     Procedure TurnRight( Src, Dst : TBitmap)
7672:     Procedure HeightMap( const Dst : TBitmap; Amount : Integer)
7673:     Procedure TexturizeTile( const Dst : TBitmap; Amount : Integer)
7674:     Procedure TexturizeOverlap( const Dst : TBitmap; Amount : Integer)
7675:     Procedure RippleRandom( const Dst : TBitmap; Amount : Integer)
7676:     Procedure RippleTooth( const Dst : TBitmap; Amount : Integer)
7677:     Procedure RippleTriangle( const Dst : TBitmap; Amount : Integer)
7678:     Procedure Triangles( const Dst : TBitmap; Amount : Integer)

```

```

7679: Procedure DrawMandelJulia(const Dst: TBitmap; x0,y0,x1,y1:Single; Niter:Integer; Mandel:Boolean)
7680: Procedure FilterXBlue( const Dst : TBitmap; Min, Max : Integer)
7681: Procedure FilterXGreen( const Dst : TBitmap; Min, Max : Integer)
7682: Procedure FilterXRed( const Dst : TBitmap; Min, Max : Integer)
7683: Procedure FilterBlue( const Dst : TBitmap; Min, Max : Integer)
7684: Procedure FilterGreen( const Dst : TBitmap; Min, Max : Integer)
7685: Procedure FilterRed( const Dst : TBitmap; Min, Max : Integer)
7686: Procedure Emboss( var Bmp : TBitmap)
7687: Procedure Plasma( Src1, Src2, Dst : TBitmap; Scale, Turbulence : Single)
7688: Procedure Shake( Src, Dst : TBitmap; Factor : Single)
7689: Procedure ShakeDown( Src, Dst : TBitmap; Factor : Single)
7690: Procedure KeepBlue( const Dst : TBitmap; Factor : Single)
7691: Procedure KeepGreen( const Dst : TBitmap; Factor : Single)
7692: Procedure KeepRed( const Dst : TBitmap; Factor : Single)
7693: Procedure Mandelbrot( const Dst : TBitmap; Factor : Integer)
7694: Procedure MaskMandelbrot( const Dst : TBitmap; Factor : Integer)
7695: Procedure FoldRight( Src1, Src2, Dst : TBitmap; Amount : Single)
7696: Procedure QuartoOpaque( Src, Dst : TBitmap)
7697: Procedure SemiOpaque( Src, Dst : TBitmap)
7698: Procedure ShadowDownLeft( const Dst : TBitmap)
7699: Procedure ShadowDownRight( const Dst : TBitmap)
7700: Procedure ShadowUpLeft( const Dst : TBitmap)
7701: Procedure ShadowUpRight( const Dst : TBitmap)
7702: Procedure Darkness( const Dst : TBitmap; Amount : Integer)
7703: Procedure Trace( const Dst : TBitmap; Intensity : Integer)
7704: Procedure FlipRight( const Dst : TBitmap)
7705: Procedure FlipDown( const Dst : TBitmap)
7706: Procedure SpotLight( const Dst : TBitmap; Amount : Integer; Spot : TRect)
7707: Procedure SplitLight( const Dst : TBitmap; Amount : Integer)
7708: Procedure MakeSeamlessClip( var Dst : TBitmap; Seam : Integer)
7709: Procedure Wave( const Dst : TBitmap; Amount, Inference, Style : Integer)
7710: Procedure Mosaic( const Bm : TBitmap; Size : Integer)
7711: Procedure SmoothRotate( var Src, Dst : TBitmap; CX, CY : Integer; Angle : Single)
7712: Procedure SmoothResize( var Src, Dst : TBitmap)
7713: Procedure Twist( var Bmp, Dst : TBitmap; Amount : Integer)
7714: Procedure SplitBlur( const Dst : TBitmap; Amount : Integer)
7715: Procedure GaussianBlur( const Dst : TBitmap; Amount : Integer)
7716: Procedure Smooth( const Dst : TBitmap; Weight : Integer)
7717: Procedure GrayScale( const Dst : TBitmap)
7718: Procedure AddColorNoise( const Dst : TBitmap; Amount : Integer)
7719: Procedure AddMonoNoise( const Dst : TBitmap; Amount : Integer)
7720: Procedure Contrast( const Dst : TBitmap; Amount : Integer)
7721: Procedure Lightness( const Dst : TBitmap; Amount : Integer)
7722: Procedure Saturation( const Dst : TBitmap; Amount : Integer)
7723: Procedure Spray( const Dst : TBitmap; Amount : Integer)
7724: Procedure AntiAlias( const Dst : TBitmap)
7725: Procedure AntiAliasRect( const Dst : TBitmap; XOrigin, YOrigin, XFinal, YFinal : Integer)
7726: Procedure SmoothPoint( const Dst : TBitmap; XK, YK : Integer)
7727: Procedure FishEye( var Bmp, Dst : TBitmap; Amount : Single)
7728: Procedure Marble( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7729: Procedure Marble2( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7730: Procedure Marble3( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7731: Procedure Marble4( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7732: Procedure Marble5( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7733: Procedure Marble6( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7734: Procedure Marble7( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7735: Procedure Marble8( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7736: Procedure SqueezeHor( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7737: Procedure SplitRound( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7738: Procedure Tile( Src, Dst : TBitmap; Amount : Integer)
7739: Procedure Stretch( Src, Dst : TBitmap; Filter : TFilterProc; AWidth : Single)
7740: Procedure Grow( Src1, Src2, Dst : TBitmap; Amount : Single; X, Y : Integer)
7741: Procedure Invert( Src : TBitmap)
7742: Procedure MirrorRight( Src : TBitmap)
7743: Procedure MirrorDown( Src : TBitmap)
7744: end;
7745: end;
7746:
7747: (*-----*)
7748: procedure SIRegister_JvPaintFX(CL: TPSPascalCompiler);
7749: begin
7750:   AddTypeS('TLightBrush', '( lbBrightness, lbContrast, lbSaturation, lbFishe'
7751:     + 'ye, lbrotate, lbtwist, lbwrist, lbbrimble, mbHor, mbBottom, mbDiamond, mbWast'
7752:     + 'e, mbRound, mbRound2, mbSplitRound, mbSplitWaste )'
7753:   );
7754:   SIRegister_TJvPaintFX(CL);
7755:   Function SplineFilter( Value : Single) : Single
7756:   Function BellFilter( Value : Single) : Single
7757:   Function TriangleFilter( Value : Single) : Single
7758:   Function BoxFilter( Value : Single) : Single
7759:   Function HermiteFilter( Value : Single) : Single
7760:   Function Lanczos3Filter( Value : Single) : Single
7761:   Function MitchellFilter( Value : Single) : Single
7762: end;
7763:
7764: (*-----*)
7765: procedure SIRegister_Chart(CL: TPSPascalCompiler);
7766: begin
7767:   'TeeMsg_DefaultFunctionName', 'String').SetString( 'TeeFunction

```

```

7768: TeeMsg_DefaultSeriesName', 'String').SetString( 'Series
7769: TeeMsg_DefaultToolName', 'String').SetString( 'ChartTool
7770: ChartComponentPalette', 'String').SetString( 'TeeChart
7771: TeeMaxLegendColumns', 'LongInt').SetInt( 2);
7772: TeeDefaultLegendSymbolWidth', 'LongInt').SetInt( 20);
7773: TeeTitleFootDistance, LongInt).SetInt( 5);
7774: SIRegister_TCustomChartWall(CL);
7775: SIRegister_TChartWall(CL);
7776: SIRegister_TChartLegendGradient(CL);
7777: TLegendStyle', '( lsAuto, lsSeries, lsValues, lsLastValues, lsSeriesGroups )
7778: TLegendAlignment', '( laLeft, laRight, laTop, laBottom )
7779: FindClass('TOBJECT'), 'LegendException
7780: TOnGetLegendText', 'Procedure ( Sender : TCustomAxisPanel; Legen'
7781: + 'dStyle : TLegendStyle; Index : Integer; var LegendText : String)
7782: FindClass('TOBJECT'), 'TCustomChartLegend
7783: TLegendSymbolSize', '( lcsPercent, lcsPixels )
7784: TLegendSymbolPosition', '( spLeft, spRight )
7785: TSymbolDrawEvent', 'Procedure(Sender:TObject;Series:TChartSeries;ValueIndex:Integer;R:TRect);
7786: TSymbolCalcHeight', 'Function : Integer
7787: SIRegister_TLegendSymbol(CL);
7788: SIRegister_TTeeCustomShapePosition(CL);
7789: TCheckBoxesStyle', '( cbsCheck, cbsRadio )
7790: SIRegister_TLegendTitle(CL);
7791: SIRegister_TLegendItem(CL);
7792: SIRegister_TLegendItems(CL);
7793: TLegendCalcSize', 'Procedure ( Sender : TCustomChartLegend; var ASize : Integer)
7794: FindClass('TOBJECT'), 'TCustomChart
7795: SIRegister_TCustomChartLegend(CL);
7796: SIRegister_TChartLegend(CL);
7797: SIRegister_TChartTitle(CL);
7798: SIRegister_TChartFootTitle(CL);
7799: TChartClick', 'Procedure ( Sender : TCustomChart; Button : TMous'
7800: + 'eButton; Shift : TShiftState; X, Y : Integer)
7801: TChartClickAxis', 'Procedure ( Sender : TCustomChart; Axis : TCh'
7802: + 'artAxis; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7803: TChartClickSeries', 'Procedure ( Sender : TCustomChart; Series : '
7804: + 'TChartSeries; ValueIndex : Integer; Button: TMouseButton; Shift:TShiftState;X,Y:Integer)
7805: TChartClickTitle', 'Procedure ( Sender : TCustomChart; ATitle : '
7806: + 'TChartTitle; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7807: TOnGetLegendPos', 'Procedure (Sender: TCustomChart; Index: Integer; var X,Y,XColor:Integer)
7808: TOnGetLegendRect', 'Procedure ( Sender : TCustomChart; var Rect : TRect)
7809: TAxisSavedScales', 'record Auto : Boolean; AutoMin : Boolean; Au'
7810: + 'toMax : Boolean; Min : Double; Max : Double; end
7811: TAllAxisSavedScales', 'array of TAxisSavedScales
7812: SIRegister_TChartBackWall(CL);
7813: SIRegister_TChartRightWall(CL);
7814: SIRegister_TChartBottomWall(CL);
7815: SIRegister_TChartLeftWall(CL);
7816: SIRegister_TChartWalls(CL);
7817: TChartAllowScrollEvent', 'Procedure(Sender:TChartAxis;var AMin,AMax:Double;var AllowScroll:Boolean);
7818: SIRegister_TCustomChart(CL);
7819: SIRegister_TChart(CL);
7820: SIRegister_TTeeSeriesTypes(CL);
7821: SIRegister_TTeeToolTypes(CL);
7822: SIRegister_TTeeDragObject(CL);
7823: SIRegister_TColorPalettes(CL);
7824: Procedure RegisterTeeSeries(ASeriesClass:TChartSeriesClass;ADescription,
AGalleryPage:PString;ANumGallerySeries:Integer;
7825: Procedure RegisterTeeSeries1( ASeriesClass : TChartSeriesClass; ADescription : PString);
7826: Procedure RegisterTeeFunction(AFunctClass:TTEEFunctionClass;ADescription,
AGalleryPage:PString;ANumGallerySeries: Int;
7827: Procedure RegisterTeeBasicFunction( AFunctionClass : TTEEFunctionClass; ADescription : PString)
7828: Procedure RegisterTeeSeriesFunction(ASeriesClass: TChartSeriesClass;AFunctionClass:TTEEFunctionClass;
ADescription, AGalleryPage: PString; ANumGallerySeries : Integer; ASubIndex : Integer)
7829: Procedure UnRegisterTeeSeries( const ASeriesList : array of TChartSeriesClass)
7830: Procedure UnRegisterTeeFunctions( const AFunctionList : array of TTEEFunctionClass)
7831: Procedure AssignSeries( var OldSeries, NewSeries : TChartSeries)
7832: Function CreateNewTeeFunction( ASeries : TChartSeries; AClass : TTEEFunctionClass ) : TTEEFunction
7833: Function CreateNewSeries( AOwner : TComponent; AChart : TCustomAxisPanel; AClass : TChartSeriesClass;
AFunctionClass : TTEEFunctionClass ) : TChartSeries
7834: Function CloneChartSeries( ASeries : TChartSeries ) : TChartSeries;
7835: Function CloneChartSeries1( ASeries : TChartSeries; AChart : TCustomAxisPanel ) : TChartSeries;
7836: Function CloneChartSeries2(ASeries:TChartSeries;AOwner:TComponent;AChart:TCustomAxisPanel):TChartSeries;;
7837: Function CloneChartTool( ATool : TTEECustomTool; AOwner : TComponent ) : TTEECustomTool
7838: Function ChangeSeriesType( var ASeries : TChartSeries; NewType : TChartSeriesClass ) : TChartSeries
7839: Procedure ChangeAllSeriesType( AChart : TCustomChart; AClass : TChartSeriesClass)
7840: Function GetNewSeriesName( AOwner : TComponent ) : TComponentName
7841: Procedure RegisterTeeTools( const ATools : array of TTEECustomToolClass)
7842: Procedure UnRegisterTeeTools( const ATools : array of TTEECustomToolClass)
7843: Function GetGallerySeriesName( ASeries : TChartSeries ) : String
7844: Procedure PaintSeriesLegend(ASeries:TChartSeries; ACanvas:TCanvas; const R:TRect;ReferenceChart:
TCustomChart);
7845: SIRegister_TChartTheme(CL);
7846: //TChartThemeClass', 'class of TChartTheme
7847: //TCanvasClass', 'class of TCanvas3D
7848: Function SeriesNameOrIndex( ASeries : TCustomChartSeries ) : String
7849: Function SeriesTitleOrName( ASeries : TCustomChartSeries ) : String
7850: Procedure FillSeriesItems( AItems : TStrings; AList : TCustomSeriesList; UseTitles : Boolean)
7851: Procedure ShowMessageUser( const S : String)

```

```

7852: Function HasNoMandatoryValues( ASeries : TChartSeries) : Boolean
7853: Function HasLabels( ASeries : TChartSeries) : Boolean
7854: Function HasColors( ASeries : TChartSeries) : Boolean
7855: Function SeriesGuessContents( ASeries : TChartSeries) : TeeFormatFlag
7856: Procedure TeeDrawBitmapEditor( Canvas : TCanvas; Element : TCustomChartElement; Left, Top : Integer)
7857: end;
7858:
7859:
7860: procedure SIRegister_TeeProcs(CL: TPSPascalCompiler);
7861: begin
7862:   //TeeFormBorderStyle', ''').SetString( bsNone);
7863:   SIRegister_TMetafile(CL);
7864:   'TeeDefVerticalMargin', 'LongInt').SetInt( 4);
7865:   'TeeDefHorizMargin', 'LongInt').SetInt( 3);
7866:   'crTeeHand', 'LongInt').SetInt( TCursor ( 2020 ));
7867:   'TeeMsg_TeeHand', 'String').SetString( 'crTeeHand
7868:   'TeeNormalPrintDetail', 'LongInt').SetInt( 0);
7869:   'TeeHighPrintDetail', 'LongInt').SetInt( - 100);
7870:   'TeeDefault_PrintMargin', 'LongInt').SetInt( 15);
7871:   'MaxDefaultColors', 'LongInt').SetInt( 19);
7872:   'TeeTabDelimiter', 'Char').SetString( #9);
7873:   TDateTimeStep', '( dtOneMicroSecond, dtOneMillisecond, dtOneSeco'
7874:   + 'nd, dtFiveSeconds, dtTenSeconds, dtFifteenSeconds, dtThirtySeconds, dtOneM'
7875:   + 'inute, dtFiveMinutes, dtTenMinutes, dtFifteenMinutes, dtThirtyMinutes, dtO'
7876:   + 'neHour, dtTwoHours, dtSixHours, dtTwelveHours, dtOneDay, dtTwoDays, dtThre'
7877:   + 'eDays, dtOneWeek, dtHalfMonth, dtOneMonth, dtTwoMonths, dtThreeMonths, dtF'
7878:   + 'ourMonths, dtSixMonths, dtOneYear, dtNone )
7879:   SIRegister_TCustomPanelNoCaption(CL);
7880:   FindClass('TObject'), 'TCustomTeePanel
7881:   SIRegister_TZoomPanning(CL);
7882:   SIRegister_TTeeEvent(CL);
7883:   //SIRegister_TTeeEventListeners(CL);
7884:   'TeeMouseEventKind', '( meDown, meUp, meMove )
7885:   SIRegister_TTeeMouseEvent(CL);
7886:   SIRegister_TCustomTeePanel(CL);
7887:   //TChartGradient', 'TeeGradient
7888:   //TChartGradientClass', 'class of TChartGradient
7889:   TPanningMode', '( pmNone, pmHorizontal, pmVertical, pmBoth )
7890:   SIRegister_TTeeZoomPen(CL);
7891:   SIRegister_TTeeZoomBrush(CL);
7892:   'TeeZoomDirection', '( tzdHorizontal, tzdVertical, tzdBoth )
7893:   SIRegister_TTeeZoom(CL);
7894:   FindClass('TObject'), 'TCustomTeePanelExtended
7895:   'TeeBackImageMode', '( pbmStretch, pbmTile, pbmCenter, pbmCustom )
7896:   SIRegister_TBackImage(CL);
7897:   SIRegister_TCustomTeePanelExtended(CL);
7898:   //TChartBrushClass', 'class of TChartBrush
7899:   SIRegister_TTeeCustomShapeBrushPen(CL);
7900:   TChartObjectShapeStyle', '( fosRectangle, fosRoundRectangle, fosEllipse )
7901:   TTextFormat', '( ttfNormal, ttfHtml )
7902:   SIRegister_TTeeCustomShape(CL);
7903:   SIRegister_TTeeShape(CL);
7904:   SIRegister_TTeeExportData(CL);
7905: Function TeeStr( const Num : Integer) : String
7906: Function DateTimeDefaultFormat( const AStep : Double) : String
7907: Function TEEDaysInMonth( Year, Month : Word) : Word
7908: Function FindDateTimeStep( const StepValue : Double) : TDateTimeStep
7909: Function NextDateTimeStep( const AStep : Double) : Double
7910: Function PointInLine( const P : TPoint; const px, py, qx, qy : Integer) : Boolean;
7911: Function PointInLine1( const P, FromPoint, ToPoint : TPoint) : Boolean;
7912: Function PointInLine2(const P,FromPoint,ToPoint:TPoint;const TolerancePixels:Integer):Boolean;
7913: Function PointInLine3( const P : TPoint; const px, py, qx, qy, TolerancePixels:Integer):Boolean;
7914: Function PointInLineTolerance(const P:TPoint;const px,py,qx,qy,TolerancePixels:Integer):Boolean;
7915: Function PointInPolygon( const P : TPoint; const Poly : array of TPoint) : Boolean
7916: Function PointInTriangle( const P, P0, P1, P2 : TPoint) : Boolean;
7917: Function PointInTriangle1( const P : TPoint; X0, X1, Y0, Y1 : Integer) : Boolean;
7918: Function PointInHorizTriangle( const P : TPoint; Y0, Y1, X0, X1 : Integer) : Boolean
7919: Function PointInEllipse( const P : TPoint; const Rect : TRect) : Boolean;
7920: Function PointInEllipse1( const P: TPoint;Left,Top,Right, Bottom : Integer) : Boolean;
7921: Function DelphiToLocalFormat( const Format : String) : String
7922: Function LocalToDelphiFormat( const Format : String) : String
7923: Procedure TeeEnableControls(Enable: Boolean; const ControlArray : array of TControl)
7924: Function TeeRoundDate(const ADate : TDateTime; AStep : TDateTimeStep) : TDateTime
7925: Procedure TeeDateTimeIncrement(IsDateTime:Boolean;Increment:Boolean;var Value:Double;const
AnIncrement:Double; tmpWhichDateTime:TDateTimeStep)
7926:   'TeeSortCompare', 'Function ( a, b : Integer) : Integer
7927:   'TeeSortSwap', 'Procedure ( a, b : Integer)
7928: Procedure TeeSort(StartIndex,EndIndex:Integer;CompareFunc:TeeSortCompare;SwapFunc:TeeSortSwap);
7929: Function TeeGetUniqueName( AOwner : TComponent; const AStartName : String) : string
7930: Function TeeExtractField( St : String; Index : Integer) : String;
7931: Function TeeExtractField1( St : String; Index : Integer; const Separator : String) : String;
7932: Function TeeNumFields( St : String) : Integer;
7933: Function TeeNumFields1( const St, Separator : String) : Integer;
7934: Procedure TeeGetBitmapEditor( AObject : TObject; var Bitmap : TBitmap)
7935: Procedure TeeLoadBitmap( Bitmap : TBitmap; const Name1, Name2 : String)
7936: // TColorArray', 'array of TColor
7937: Function GetDefaultColor( const Index : Integer) : TColor
7938: Procedure SetDefaultColorPalette;
7939: Procedure SetDefaultColorPalette1( const Palette : array of TColor);

```



```

7940: 'TeeCheckBoxSize','LongInt').SetInt( 11);
7941: Procedure TeeDrawCheckBox(x,y:Integer;Canvas:TCanvas;Checked:Boolean;ABackColor:TColor;CheckBox:Boolean);
7942: Function TEEStrToFloatDef( const S : string; const Default : Extended ) : Extended
7943: Function TryStrToFloat( const S : String; var Value : Double ) : Boolean
7944: Function CrossingLines( const X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Double; out x, y : Double ) : Boolean
7945: Procedure TeeTranslateControl( AControl : TControl);
7946: Procedure TeeTranslateControl1( AControl : TControl; const ExcludeChilds : array of TControl);
7947: Function ReplaceChar( const AString : String; const Search : Char; const Replace : Char ) : String
7948: //Procedure RectToFourPoints( const ARect : TRect; const Angle : Double; var P : TFourPoints)
7949: Function TeeAntiAlias( Panel : TCustomTeePanel; ChartRect : Boolean ) : TBitmap
7950: //Procedure DrawBevel(Canvas:TCanvas;Bevel:TPanelBevel;var R:TRect;Width:Integer;Round:Integer);
7951: //Function ScreenRatio( ACanvas : TCanvas3D ) : Double
7952: Function TeeReadBoolOption( const AKey : String; DefaultValue : Boolean ) : Boolean
7953: Procedure TeeSaveBoolOption( const AKey : String; Value : Boolean)
7954: Function TeeReadIntegerOption( const AKey : String; DefaultValue : Integer ) : Integer
7955: Procedure TeeSaveIntegerOption( const AKey : String; Value : Integer)
7956: Function TeeReadStringOption( const AKey, DefaultValue : String ) : String
7957: Procedure TeeSaveStringOption( const AKey, Value : String)
7958: Function TeeDefaultXMLEncoding : String
7959: Procedure ConvertTextToXML( Stream : TStream; XMLHeader : Boolean)
7960: TeeWindowHandle, 'Integer
7961: Procedure TeeGotoURL( Handle : TeeWindowHandle; const URL : String)
7962: Procedure HtmlTextOut( ACanvas : TCanvas; x, y : Integer; Text : String)
7963: Function HtmlTextExtent( ACanvas : TCanvas; const Text : String ) : TSize
7964: end;
7965:
7966:
7967: using mXBDEUtils
7968: *****
7969: Procedure SetAlias( aAlias, aDirectory : String)
7970: Procedure CheckRegistryEntry(Reg:TRegistry;const Path,Value:String;const Default,
Desired:Variant;Size:Byte);
7971: Function GetFileVersionNumber( const FileName : String ) : TVersionNo
7972: Procedure SetBDE( aPath, aNode, aValue : String)
7973: function RestartDialog(Wnd: HWND; Reason: PChar; Flags: Integer): Integer; stdcall;
7974: Function GetSystemDirectory( lpBuffer : string; uSize : UINT ) : UINT
7975: Function GetSystemDirectoryW( lpBuffer : pchar; uSize : UINT ) : UINT
7976: Function GetTempPath( nBufferLength : DWORD; lpBuffer : string ) : DWORD
7977: Function GetWindowsDirectoryW( nBufferLength : DWORD; lpBuffer : string ) : DWORD
7978: Function GetTempFileName(lpPathName,lpPrefixString:string;uUnique:UINT;lpTempFileName:string):UINT;
7979:
7980:
7981: procedure SIRegister_cDateTime(CL: TPSPascalCompiler);
7982: begin
7983: AddClassN(FindClass('TOBJECT'),'EDateTime
7984: Function DatePart( const D : TDateTime ) : Integer
7985: Function TimePart( const D : TDateTime ) : Double
7986: Function Century( const D : TDateTime ) : Word
7987: Function Year( const D : TDateTime ) : Word
7988: Function Month( const D : TDateTime ) : Word
7989: Function Day( const D : TDateTime ) : Word
7990: Function Hour( const D : TDateTime ) : Word
7991: Function Minute( const D : TDateTime ) : Word
7992: Function Second( const D : TDateTime ) : Word
7993: Function Millisecond( const D : TDateTime ) : Word
7994: ('OneDay','Extended').setExtended( 1.0);
7995: ('OneHour','Extended').SetExtended( OneDay / 24);
7996: ('OneMinute','Extended').SetExtended( OneHour / 60);
7997: ('OneSecond','Extended').SetExtended( OneMinute / 60);
7998: ('OneMillisecond','Extended').SetExtended( OneSecond / 1000);
7999: ('OneWeek','Extended').SetExtended( OneDay * 7);
8000: ('HoursPerDay','Extended').SetExtended( 24);
8001: ('MinutesPerHour','Extended').SetExtended( 60);
8002: ('SecondsPerMinute','Extended').SetExtended( 60);
8003: Procedure SetYear( var D : TDateTime; const Year : Word)
8004: Procedure SetMonth( var D : TDateTime; const Month : Word)
8005: Procedure SetDay( var D : TDateTime; const Day : Word)
8006: Procedure SetHour( var D : TDateTime; const Hour : Word)
8007: Procedure SetMinute( var D : TDateTime; const Minute : Word)
8008: Procedure SetSecond( var D : TDateTime; const Second : Word)
8009: Procedure SetMillisecond( var D : TDateTime; const Milliseconds : Word)
8010: Function IsEqual( const D1, D2 : TDateTime ) : Boolean;
8011: Function IsEqual1( const D1 : TDateTime; const Ye, Da : Word):Boolean;
8012: Function IsEqual2( const D1 : TDateTime; const Ho, Mi, Se, ms : Word):Boolean;
8013: Function IsAM( const D : TDateTime ) : Boolean
8014: Function IsPM( const D : TDateTime ) : Boolean
8015: Function IsMidnight( const D : TDateTime ) : Boolean
8016: Function IsNoon( const D : TDateTime ) : Boolean
8017: Function IsSunday( const D : TDateTime ) : Boolean
8018: Function IsMonday( const D : TDateTime ) : Boolean
8019: Function IsTuesday( const D : TDateTime ) : Boolean
8020: Function IsWednesday( const D : TDateTime ) : Boolean
8021: Function IsThursday( const D : TDateTime ) : Boolean
8022: Function IsFriday( const D : TDateTime ) : Boolean
8023: Function IsSaturday( const D : TDateTime ) : Boolean
8024: Function IsWeekend( const D : TDateTime ) : Boolean
8025: Function Noon( const D : TDateTime ) : TDateTime
8026: Function Midnight( const D : TDateTime ) : TDateTime
8027: Function FirstDayOfMonth( const D : TDateTime ) : TDateTime

```

```

8028: Function LastDayOfMonth( const D : TDateTime) : TDateTime
8029: Function NextWorkday( const D : TDateTime) : TDateTime
8030: Function PreviousWorkday( const D : TDateTime) : TDateTime
8031: Function FirstDayOfYear( const D : TDateTime) : TDateTime
8032: Function LastDayOfYear( const D : TDateTime) : TDateTime
8033: Function EasterSunday( const Year : Word) : TDateTime
8034: Function GoodFriday( const Year : Word) : TDateTime
8035: Function AddMilliseconds( const D : TDateTime; const N : Int64) : TDateTime
8036: Function AddSeconds( const D : TDateTime; const N : Int64) : TDateTime
8037: Function AddMinutes( const D : TDateTime; const N : Integer) : TDateTime
8038: Function AddHours( const D : TDateTime; const N : Integer) : TDateTime
8039: Function AddDays( const D : TDateTime; const N : Integer) : TDateTime
8040: Function AddWeeks( const D : TDateTime; const N : Integer) : TDateTime
8041: Function AddMonths( const D : TDateTime; const N : Integer) : TDateTime
8042: Function AddYears( const D : TDateTime; const N : Integer) : TDateTime
8043: Function DayOfYear( const Ye, Mo, Da : Word) : Integer
8044: Function DayOfYear( const D : TDateTime) : Integer
8045: Function DaysInMonth( const Ye, Mo : Word) : Integer
8046: Function DaysInMonth( const D : TDateTime) : Integer
8047: Function DaysInYear( const Ye : Word) : Integer
8048: Function DaysInYearDate( const D : TDateTime) : Integer
8049: Function WeekNumber( const D : TDateTime) : Integer
8050: Function ISOFirstWeekOfYear( const Ye : Word) : TDateTime
8051: Procedure ISOWeekNumber( const D : TDateTime; var WeekNumber, WeekYear : Word)
8052: Function DiffMilliseconds( const D1, D2 : TDateTime) : Int64
8053: Function DiffSeconds( const D1, D2 : TDateTime) : Integer
8054: Function DiffMinutes( const D1, D2 : TDateTime) : Integer
8055: Function DiffHours( const D1, D2 : TDateTime) : Integer
8056: Function DiffDays( const D1, D2 : TDateTime) : Integer
8057: Function DiffWeeks( const D1, D2 : TDateTime) : Integer
8058: Function DiffMonths( const D1, D2 : TDateTime) : Integer
8059: Function DiffYears( const D1, D2 : TDateTime) : Integer
8060: Function GMTBias : Integer
8061: Function GMTTimeToLocalTime( const D : TDateTime) : TDateTime
8062: Function LocalTimeToGMTTime( const D : TDateTime) : TDateTime
8063: Function NowAsGMTTime : TDateTime
8064: Function DateTimeToISO8601String( const D : TDateTime) : AnsiString
8065: Function ISO8601StringToTime( const D : AnsiString) : TDateTime
8066: Function ISO8601StringAsDateTime( const D : AnsiString) : TDateTime
8067: Function DateTimeToANSI( const D : TDateTime) : Integer
8068: Function ANSIToDateTime( const Julian : Integer) : TDateTime
8069: Function DateTimeToISOInteger( const D : TDateTime) : Integer
8070: Function DateTimeToISOString( const D : TDateTime) : AnsiString
8071: Function ISOIntegerToDateTime( const ISOInteger : Integer) : TDateTime
8072: Function TwoDigitRadix2000YearToYear( const Y : Integer) : Integer
8073: Function DateTimeAsElapsedTime( const D: TDateTime; const IncludeMilliseconds: Boolean): AnsiString
8074: Function UnixTimeToDateTime( const UnixTime : LongWord) : TDateTime
8075: Function DateTimeToUnixTime( const D : TDateTime) : LongWord
8076: Function EnglishShortDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8077: Function EnglishShortDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8078: Function EnglishLongDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8079: Function EnglishLongDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8080: Function EnglishShortMonthStrA( const Month : Integer) : AnsiString
8081: Function EnglishShortMonthStrU( const Month : Integer) : UnicodeString
8082: Function EnglishLongMonthStrA( const Month : Integer) : AnsiString
8083: Function EnglishLongMonthStrU( const Month : Integer) : UnicodeString
8084: Function EnglishShortDayOfWeekA( const S : AnsiString) : Integer
8085: Function EnglishShortDayOfWeekU( const S : UnicodeString) : Integer
8086: Function EnglishLongDayOfWeekA( const S : AnsiString) : Integer
8087: Function EnglishLongDayOfWeekU( const S : UnicodeString) : Integer
8088: Function EnglishShortMonthA( const S : AnsiString) : Integer
8089: Function EnglishShortMonthU( const S : UnicodeString) : Integer
8090: Function EnglishLongMonthA( const S : AnsiString) : Integer
8091: Function EnglishLongMonthU( const S : UnicodeString) : Integer
8092: Function RFC850DayOfWeekA( const S : AnsiString) : Integer
8093: Function RFC850DayOfWeekU( const S : UnicodeString) : Integer
8094: Function RFC1123DayOfWeekA( const S : AnsiString) : Integer
8095: Function RFC1123DayOfWeekU( const S : UnicodeString) : Integer
8096: Function RFCMonthA( const S : AnsiString) : Word
8097: Function RFCMonthU( const S : UnicodeString) : Word
8098: Function GMTTimeToRFC1123TimeA( const D : TDateTime; const IncludeSeconds: Boolean) : AnsiString
8099: Function GMTTimeToRFC1123TimeU( const D : TDateTime; const IncludeSeconds: Boolean) : UnicodeString
8100: Function GMTDateTimeToRFC1123DateTimeA( const D: TDateTime; const IncludeDayOfWeek: Bool): AnsiString;
8101: Function GMTDateTimeToRFC1123DateTimeU( const D: TDateTime; const IncludeDayOfWeek: Bool): UnicodeString;
8102: Function DateTimeToRFCDateTimeA( const D : TDateTime) : AnsiString
8103: Function DateTimeToRFCDateTimeU( const D : TDateTime) : UnicodeString
8104: Function NowAsRFCDateTimeA : AnsiString
8105: Function NowAsRFCDateTimeU : UnicodeString
8106: Function RFCDateTimeToGMTDateTime( const S : AnsiString) : TDateTime
8107: Function RFCDateTimeToDateTime( const S : AnsiString) : TDateTime
8108: Function RFCTimeZoneToGMTBias( const Zone : AnsiString) : Integer
8109: Function TimePeriodStr( const D : TDateTime) : AnsiString
8110: Procedure SelfTest
8111: end;
8112: *****CFileUtils
8113: Function PathHasDriveLetterA( const Path : AnsiString) : Boolean
8114: Function PathHasDriveLetter( const Path : String) : Boolean
8115: Function PathIsDriveLetterA( const Path : AnsiString) : Boolean
8116: Function PathIsDriveLetter( const Path : String) : Boolean

```

```

8117: Function PathIsDriveRootA( const Path : AnsiString ) : Boolean
8118: Function PathIsDriveRoot( const Path : String ) : Boolean
8119: Function PathIsRootA( const Path : AnsiString ) : Boolean
8120: Function PathIsRoot( const Path : String ) : Boolean
8121: Function PathIsUNCPathA( const Path : AnsiString ) : Boolean
8122: Function PathIsUNCPath( const Path : String ) : Boolean
8123: Function PathIsAbsoluteA( const Path : AnsiString ) : Boolean
8124: Function PathIsAbsolute( const Path : String ) : Boolean
8125: Function PathIsDirectoryA( const Path : AnsiString ) : Boolean
8126: Function PathIsDirectory( const Path : String ) : Boolean
8127: Function PathInclSuffixA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8128: Function PathInclSuffix( const Path : String; const PathSep : Char ) : String
8129: Function PathExclSuffixA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8130: Function PathExclSuffix( const Path : String; const PathSep : Char ) : String
8131: Procedure PathEnsureSuffixA( var Path : AnsiString; const PathSep : Char )
8132: Procedure PathEnsureSuffix( var Path : String; const PathSep : Char )
8133: Procedure PathEnsureNoSuffixA( var Path : AnsiString; const PathSep : Char )
8134: Procedure PathEnsureNoSuffix( var Path : String; const PathSep : Char )
8135: //Function PathCanonicalA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8136: Function PathCanonical( const Path : String; const PathSep : Char ) : String
8137: Function PathExpandA( const Path:AnsiString; const BasePath:AnsiString; const PathSep:Char ):AnsiString
8138: Function PathExpand( const Path : String; const BasePath : String; const PathSep : Char ) : String
8139: Function PathLeftElementA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8140: Function PathLeftElement( const Path : String; const PathSep : Char ) : String
8141: Procedure PathSplitLeftElementA( const Path:AString; var LeftElement,RightPath:AString; const PathSep:Char );
8142: Procedure PathSplitLeftElement( const Path:String; var LeftElement,RightPath : String; const PathSep:Char );
8143: Procedure DecodeFilePathA( const FilePath:AnsiString; var Path,FileName:AnsiString; const PathSep:Char;
8144: Procedure DecodeFilePath( const FilePath : String; var Path, FileName : String; const PathSep : Char )
8145: Function FileNameValidA( const FileName : AnsiString ) : AnsiString
8146: Function FileNameValid( const FileName : String ) : String
8147: Function FilePathA( const FileName,Path:AnsiString; const BasePath:AnsiStr; const PathSep:Char ):AnsiString;
8148: Function FilePath( const FileName, Path : String; const BasePath : String; const PathSep : Char ) : String
8149: Function DirectoryExpandA( const Path:AnsiString; const BasePath:AnsiString; const PathSep:Char ):AnsiString
8150: Function DirectoryExpand( const Path : String; const BasePath : String; const PathSep : Char ) : String
8151: Function UnixPathToWinPath( const Path : AnsiString ) : AnsiString
8152: Function WinPathToUnixPath( const Path : AnsiString ) : AnsiString
8153: Procedure CCopyFile( const FileName, DestName : String )
8154: Procedure CMoveFile( const FileName, DestName : String )
8155: Function CDeleteFiles( const FileMask : String ) : Boolean
8156: Function FileSeekEx( const FHandle:TFileHandle; const FileOffset:Int64; const FilePos:TFileSeekPos ):Int64;
8157: Procedure FileCloseEx( const FileHandle : TFileHandle )
8158: Function FileExistsA( const FileName : AnsiString ) : Boolean
8159: Function CFileExists( const FileName : String ) : Boolean
8160: Function CFileGetSize( const FileName : String ) : Int64
8161: Function FileGetDateTime( const FileName : String ) : TDateTime
8162: Function FileGetDateTime2( const FileName : String ) : TDateTime
8163: Function FileIsReadOnly( const FileName : String ) : Boolean
8164: Procedure FileDeleteEx( const FileName : String )
8165: Procedure FileRenameEx( const OldFileName, NewFileName : String )
8166: Function ReadFileStrA( const FileName:AnsiString; const FileSharing : TFileSharing; const FileCreationMode
: TFileCreationMode; const FileOpenWait : PFileOpenWait ) : AnsiString
8167: Function DirectoryEntryExists( const Name : String ) : Boolean
8168: Function DirectoryEntrySize( const Name : String ) : Int64
8169: Function CDirectoryExists( const DirectoryName : String ) : Boolean
8170: Function DirectoryGetDateTime( const DirectoryName : String ) : TDateTime
8171: Procedure CDirectoryCreate( const DirectoryName : String )
8172: Function GetFirstFileNameMatching( const FileMask : String ) : String
8173: Function DirEntryGetAttr( const FileName : AnsiString ) : Integer
8174: Function DirEntryIsDirectory( const FileName : AnsiString ) : Boolean
8175: Function FileHasAttr( const FileName : String; const Attr : Word ) : Boolean
8176: AddTypeS('TLogicalDriveType', '( DriveRemovable, DriveFixed, DriveRemote, '
8177: + 'DriveCDRom, DriveRamDisk, DriveTypeUnknown )
8178: Function DriveIsValid( const Drive : Char ) : Boolean
8179: Function DriveGetType( const Path : AnsiString ) : TLogicalDriveType
8180: Function DriveFreeSpace( const Path : AnsiString ) : Int64
8181:
8182: procedure SIRegister_cTimers(CL: TPSPascalCompiler);
8183: begin
8184:   AddClassN(FindClass('TOBJECT'),'ETimers
8185:   Const('TickFrequency','LongInt').SetInt( 1000 );Function GetTick : LongWord
8186: Function TickDelta( const D1, D2 : LongWord ) : Integer
8187: Function TickDeltaW( const D1, D2 : LongWord ) : LongWord
8188:   AddTypeS('THPTimer', 'Int64
8189: Procedure StartTimer( var Timer : THPTimer )
8190: Procedure StopTimer( var Timer : THPTimer )
8191: Procedure ResumeTimer( var StoppedTimer : THPTimer )
8192: Procedure InitStoppedTimer( var Timer : THPTimer )
8193: Procedure InitElapsedTimer( var Timer : THPTimer; const Milliseconds : Integer )
8194: Function MillisecondsElapsed( const Timer : THPTimer; const TimerRunning : Boolean ) : Integer
8195: Function MicrosecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean ) : Int64
8196: Procedure WaitMicroseconds( const MicroSeconds : Integer )
8197: Function GetHighPrecisionFrequency : Int64
8198: Function GetHighPrecisionTimerOverhead : Int64
8199: Procedure AdjustTimerForOverhead( var StoppedTimer : THPTimer; const Overhead : Int64 )
8200: Procedure SelfTestCTimer
8201: end;
8202:
8203: procedure SIRegister_cRandom(CL: TPSPascalCompiler);
8204: begin

```

```

8205: Function RandomSeed : LongWord
8206: Procedure AddEntropy( const Value : LongWord)
8207: Function RandomUniform : LongWord;
8208: Function RandomUniform1( const N : Integer) : Integer;
8209: Function RandomBoolean : Boolean
8210: Function RandomByte : Byte
8211: Function RandomByteNonZero : Byte
8212: Function RandomWord : Word
8213: Function RandomInt64 : Int64;
8214: Function RandomInt64( const N : Int64) : Int64;
8215: Function RandomHex( const Digits : Integer) : String
8216: Function RandomFloat : Extended
8217: Function RandomAlphaStr( const Length : Integer) : AnsiString
8218: Function RandomPseudoWord( const Length : Integer) : AnsiString
8219: Function RandomPassword(const MinL,MaxLength:Int;const CaseSens,UseSymbols,UseNumbers:Bool):AnsiString;
8220: Function mwcRandomLongWord : LongWord
8221: Function urnRandomLongWord : LongWord
8222: Function moaRandomFloat : Extended
8223: Function mwcRandomFloat : Extended
8224: Function RandomNormalF : Extended
8225: Procedure SelfTestCRRandom
8226: end;
8227:
8228: procedure SIRegister_SynEditMiscProcs(CL: TPSPascalCompiler);
8229: begin
8230: // PIntArray', 'TIntArray // will not work
8231: AddTypes('TConvertTabsProc','function(const Line:AnsiString; TabWidth: integer):AnsiString
8232: AddTypes('TConvertTabsProcEx','function(const Line:AnsiString; TabWidth: integer;var HasTabs: boolean):
AnsiString
8233: Function synMax( x, y : integer) : integer
8234: Function synMin( x, y : integer) : integer
8235: Function synMinMax( x, mi, ma : integer) : integer
8236: Procedure synSwapInt( var l, r : integer)
8237: Function synMaxPoint( const P1, P2 : TPoint) : TPoint
8238: Function synMinPoint( const P1, P2 : TPoint) : TPoint
8239: //Function synGetIntArray( Count : Cardinal; InitialValue : integer) : PIntArray
8240: Procedure synInternalFillRect( dc : HDC; const rcPaint : TRect)
8241: Function synGetBestConvertTabsProc( TabWidth : integer) : TConvertTabsProc
8242: Function synConvertTabs( const Line : AnsiString; TabWidth : integer) : AnsiString
8243: Function synGetBestConvertTabsProcEx( TabWidth : integer) : TConvertTabsProcEx
8244: Function synConvertTabsEx(const Line:AnsiString;TabWidth:integer; var HasTabs:boolean):AnsiString;
8245: Function synGetExpandedLength( const aStr : string; aTabWidth : integer) : integer
8246: Function synCharIndex2CaretPos( Index, TabWidth : integer; const Line : string) : integer
8247: Function synCaretPos2CharIndex(Position,TabWidth:int;const Line:string;var InsideTabChar:boolean):int;
8248: Function synStrScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8249: Function synStrRScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8250: TStringType', '( stNone, stHalfNumAlpha, stHalfSymbol, stHalfKat'
8251: + 'akana, stWideNumAlpha, stWideSymbol, stWideKatakana, stHiragana, stIdeograph, stControl, stKashida )
8252: ('C3_NONSPACING', 'LongInt').SetInt( 1);
8253: ('C3_DIACRITIC', 'LongInt').SetInt( 2);
8254: ('C3_VOWELMARK', 'LongInt').SetInt( 4);
8255: ('C3_SYMBOL', 'LongInt').SetInt( 8);
8256: ('C3_KATAKANA', 'LongWord').SetUInt( $0010);
8257: ('C3_HIRAGANA', 'LongWord').SetUInt( $0020);
8258: ('C3_HALFWIDTH', 'LongWord').SetUInt( $0040);
8259: ('C3_FULLWIDTH', 'LongWord').SetUInt( $0080);
8260: ('C3_IDEOGRAPH', 'LongWord').SetUInt( $0100);
8261: ('C3_KASHIDA', 'LongWord').SetUInt( $0200);
8262: ('C3_LEXICAL', 'LongWord').SetUInt( $0400);
8263: ('C3_ALPHA', 'LongWord').SetUInt( $8000);
8264: ('C3_NOTAPPLICABLE', 'LongInt').SetInt( 0);
8265: Function synStrScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8266: Function synStrRScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8267: Function synIsStringType( Value : Word) : TStringType
8268: Function synGetEOL( Line : PChar) : PChar
8269: Function synEncodeString( s : string) : string
8270: Function synDecodeString( s : string) : string
8271: Procedure synFreeAndNil( var Obj: TObject)
8272: Procedure synAssert( Expr : Boolean)
8273: Function synLastDelimiter( const Delimiters, S : string) : Integer
8274: TReplaceFlag', '( rfReplaceAll, rfIgnoreCase )
8275: TReplaceFlags', 'set of TReplaceFlag )
8276: Function synStringReplace(const S, OldPattern, NewPattern : string; Flags: TReplaceFlags) : string
8277: Function synGetRValue( RGBValue : TColor) : byte
8278: Function synGetGValue( RGBValue : TColor) : byte
8279: Function synGetBValue( RGBValue : TColor) : byte
8280: Function synRGB( r, g, b : Byte) : Cardinal
8281: // THighlighterAttriProc', 'Function ( Highlighter : TSynCustomHigh'
8282: // + 'lighter; Attri:TSynHighlighterAttributes;UniqueAttriName:string;Params array of Pointer):Boolean;
8283: //Function synEnumHighlighterAttris( Highlighter : TSynCustomHighlighter; SkipDuplicates : Boolean;
HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer) : Boolean
8284: Function synCalcFCS( const ABuf, ABufSize : Cardinal) : Word
8285: Procedure synSynDrawGradient(const ACanvas:TCanvas;const AStartColor,
AEndColor:TColor;ASteps:integer;const ARect : TRect; const AHorizontal : boolean)
8286: end;
8287:
8288: Function GET_APPCOMMAND_LPARAM( lParam : LPARAM) : WORD
8289: Function GET_DEVICE_LPARAM( lParam : LPARAM) : WORD
8290: Function GET_KEYSTATE_LPARAM( lParam : LPARAM) : WORD

```



```

8291:
8292: procedure SIRegister_synautil(CL: TPSPascalCompiler);
8293: begin
8294:   Function STimeZoneBias : integer
8295:   Function TimeZone : string
8296:   Function Rfc822DateTime( t : TDateTime) : string
8297:   Function CDateTime( t : TDateTime) : string
8298:   Function SimpleDateTime( t : TDateTime) : string
8299:   Function AnsiCDateTime( t : TDateTime) : string
8300:   Function GetMonthNumber( Value : String) : integer
8301:   Function GetTimeFromStr( Value : string) : TDateTime
8302:   Function GetDateMDYFromStr( Value : string) : TDateTime
8303:   Function DecodeRfcDateTime( Value : string) : TDateTime
8304:   Function GetUTTime : TDateTime
8305:   Function SetUTTime( Newdt : TDateTime) : Boolean
8306:   Function SGetTick : LongWord
8307:   Function STickDelta( TickOld, TickNew : LongWord) : LongWord
8308:   Function CodeInt( Value : Word) : Ansistring
8309:   Function DecodeInt( const Value : Ansistring; Index : Integer) : Word
8310:   Function CodeLongInt( Value : LongInt) : Ansistring
8311:   Function DecodeLongInt( const Value : Ansistring; Index : Integer) : LongInt
8312:   Function DumpStr( const Buffer : Ansistring) : string
8313:   Function DumpExStr( const Buffer : Ansistring) : string
8314:   Procedure Dump( const Buffer : Ansistring; DumpFile : string)
8315:   Procedure DumpEx( const Buffer : Ansistring; DumpFile : string)
8316:   Function TrimSPLeft( const S : string) : string
8317:   Function TrimSPRight( const S : string) : string
8318:   Function TrimSP( const S : string) : string
8319:   Function SeparateLeft( const Value, Delimiter : string) : string
8320:   Function SeparateRight( const Value, Delimiter : string) : string
8321:   Function SGetParameter( const Value, Parameter : string) : string
8322:   Procedure ParseParametersEx( Value, Delimiter : string; const Parameters : TStrings)
8323:   Procedure ParseParameters( Value : string; const Parameters : TStrings)
8324:   Function IndexByBegin( Value : string; const List : TStrings) : integer
8325:   Function GetEmailAddr( const Value : string) : string
8326:   Function GetEmailDesc( Value : string) : string
8327:   Function CStrToHex( const Value : Ansistring) : string
8328:   Function CIntToBin( Value : Integer; Digits : Byte) : string
8329:   Function CBinToInt( const Value : string) : Integer
8330:   Function ParseURL( URL : string; var Prot, User, Pass, Host, Port, Path, Para:string):string
8331:   Function CReplaceString( Value, Search, Replace : Ansistring) : Ansistring
8332:   Function CRPosEx( const Sub, Value : string; From : integer) : Integer
8333:   Function CRPos( const Sub, Value : String) : Integer
8334:   Function FetchBin( var Value : string; const Delimiter : string) : string
8335:   Function CFetch( var Value : string; const Delimiter : string) : string
8336:   Function FetchEx( var Value : string; const Delimiter, Quotation : string) : string
8337:   Function IsBinaryString( const Value : Ansistring) : Boolean
8338:   Function PosCRLF( const Value : Ansistring; var Terminator : Ansistring) : integer
8339:   Procedure StringsTrim( const value : TStrings)
8340:   Function PosFrom( const SubStr, Value : String; From : integer) : integer
8341:   Function IncPoint( const p : __pointer; Value : integer) : __pointer
8342:   Function GetBetween( const PairBegin, PairEnd, Value : string) : string
8343:   Function CCountOfChar( const Value : string; aChr : char) : integer
8344:   Function UnquoteStr( const Value : string; Quote : Char) : string
8345:   Function QuoteStr( const Value : string; Quote : Char) : string
8346:   Procedure HeadersToList( const Value : TStrings)
8347:   Procedure ListToHeaders( const Value : TStrings)
8348:   Function SwapBytes( Value : integer) : integer
8349:   Function ReadStrFromStream( const Stream : TStream; len : integer) : Ansistring
8350:   Procedure WriteStrToStream( const Stream : TStream; Value : Ansistring)
8351:   Function GetTempFile( const Dir, prefix : Ansistring) : Ansistring
8352:   Function CPadString( const Value : Ansistring; len : integer; Pad : AnsiChar) : Ansistring
8353:   Function CXorString( Indatal, Indata2 : Ansistring) : Ansistring
8354:   Function NormalizeHeader( Value : TStrings; var Index : Integer) : string
8355: end;
8356:
8357: procedure SIRegister_StCRC(CL: TPSPascalCompiler);
8358: begin
8359:   ('CrcBufSize','LongInt').SetInt( 2048);
8360:   Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8361:   Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8362:   Function Adler32OfFile( FileName : Ansistring) : LongInt
8363:   Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8364:   Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8365:   Function Crc16OfFile( FileName : Ansistring) : Cardinal
8366:   Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8367:   Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8368:   Function Crc32OfFile( FileName : Ansistring) : LongInt
8369:   Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8370:   Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8371:   Function InternetSumOfFile( FileName : Ansistring) : Cardinal
8372:   Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8373:   Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8374:   Function Kermit16OfFile( FileName : Ansistring) : Cardinal
8375: end;
8376:
8377: procedure SIRegister_ComObj(cl: TPSPascalCompiler);
8378: begin
8379:   function CreateOleObject(const ClassName : String) : IDispatch;

```

```

8380: function GetActiveOleObject(const ClassName: String): IDispatch;
8381: function ProgIDToClassID(const ProgID: string): TGUID;
8382: function ClassIDToProgID(const ClassID: TGUID): string;
8383: function CreateClassID: string;
8384: function CreateGUIDString: string;
8385: function CreateGUIDID: string;
8386: procedure OleError(ErrorCode: longint);
8387: procedure OleCheck(Result: HRESULT);
8388: end;
8389:
8390: Function xCreateOleObject( const ClassName : string) : Variant //or IDispatch
8391: Function xGetActiveOleObject( const ClassName : string) : Variant
8392: //Function DllGetClassObject( const CLSID : TCLSID; const IID : TIID; var Obj) : HRESULT
8393: Function DllCanUnloadNow : HRESULT
8394: Function DllRegisterServer : HRESULT
8395: Function DllUnregisterServer : HRESULT
8396: Function VarFromInterface( Unknown : IUnknown) : Variant
8397: Function VarToInterface( const V : Variant) : IDispatch
8398: Function VarToAutoObject( const V : Variant) : TAutoObject
8399: //Procedure
DispInvoke(Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer;Res:PVariant);
8400: //Procedure DispInvokeError( Status : HRESULT; const ExcepInfo : TExcepInfo)
8401: Procedure OleError( ErrorCode : HRESULT)
8402: Procedure OleCheck( Result : HRESULT)
8403: Function StringToClassID( const S : string) : TCLSID
8404: Function ClassIDToString( const ClassID : TCLSID) : string
8405: Function xProgIDToClassID( const ProgID : string) : TCLSID
8406: Function xClassIDToProgID( const ClassID : TCLSID) : string
8407: Function xWideCompareStr( const S1, S2 : WideString) : Integer
8408: Function xWideSameStr( const S1, S2 : WideString) : Boolean
8409: Function xGUIDToString( const ClassID : TGUID) : string
8410: Function xStringToGUID( const S : string) : TGUID
8411: Function xGetModuleName( Module : HMODULE) : string
8412: Function xAcquireExceptionObject : TObject
8413: Function xIfThen( AValue : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer
8414: Function xUtf8Encode( const WS : WideString) : UTF8String
8415: Function xUtf8Decode( const S : UTF8String) : WideString
8416: Function xExcludeTrailingPathDelimiter( const S : string) : string
8417: Function xIncludeTrailingPathDelimiter( const S : string) : string
8418: Function XRTLHandleCOMException : HRESULT
8419: Procedure XRTLCheckArgument( Flag : Boolean)
8420: //Procedure XRTLCheckOutArgument( out Arg)
8421: Procedure XRTLInterfaceConnect(const Source:IUnknown;const IID:TIID;const Sink:IUnknown;var
Connection:Longint);
8422: Procedure XRTLInterfaceDisconnect(const Source: IUnknown; const IID:TIID;var Connection : Longint)
8423: Function XRTLRegisterActiveObject(const Unk:IUnknown;ClassID:TCLSID;Flags:DWORD;var
RegisterCookie:Int):HRESULT
8424: Function XRTLUnRegisterActiveObject( var RegisterCookie : Integer) : HRESULT
8425: //Function XRTLGetActiveObject( ClassID : TCLSID; RIID : TIID; out Obj) : HRESULT
8426: Procedure XRTLEnumActiveObjects( Strings : TStrings)
8427: function XRTLDefaultCategoryManager: IUnknown;
8428: function XRTLIsCategoryEmpty(CatID: TGUID; const CategoryManager: IUnknown = nil): Boolean;
8429: // ICatRegister helper functions
8430: function XRTLCreateComponentCategory(CatID: TGUID; CatDescription: WideString;
8431: LocaleID: TLCID = LOCALE_USER_DEFAULT;
8432: const CategoryManager: IUnknown = nil): HRESULT;
8433: function XRTLRemoveComponentCategory(CatID: TGUID; CatDescription: WideString;
8434: LocaleID: TLCID = LOCALE_USER_DEFAULT;
8435: const CategoryManager: IUnknown = nil): HRESULT;
8436: function XRTLRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8437: const CategoryManager: IUnknown = nil): HRESULT;
8438: function XRTLUnRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8439: const CategoryManager: IUnknown = nil): HRESULT;
8440: // ICatInformation helper functions
8441: function XRTLGetCategoryDescription(CatID: TGUID; var CatDescription: WideString;
8442: LocaleID: TLCID = LOCALE_USER_DEFAULT;
8443: const CategoryManager: IUnknown = nil): HRESULT;
8444: function XRTLGetCategoryList(Strings: TStrings; LocaleID: TLCID = LOCALE_USER_DEFAULT;
8445: const CategoryManager: IUnknown = nil): HRESULT;
8446: function XRTLGetCategoryCLSIDList(CatID: TGUID; Strings: TStrings;
8447: const CategoryManager: IUnknown = nil): HRESULT;
8448: function XRTLGetCategoryProgIDList(CatID: TGUID; Strings: TStrings;
8449: const CategoryManager: IUnknown = nil): HRESULT;
8450: function XRTLFetch(var AInput: WideString; const ADelim: WideString = ' ';
8451: const ADelete: Boolean = True): WideString;
8452: function XRTLRPos(const ASub, AIn: WideString; AStart: Integer = -1): Integer;
8453: Function XRTLGetVariantAsString( const Value : Variant) : string
8454: Function XRTLDateTimeToTimeZoneTime( DT : TDateTime; TimeZone : TXRTLTimeZone) : TDateTime
8455: Function XRTLGetTimeZones : TXRTLTimeZones
8456: Function XFileTimeToDateTime( FileTime : TFileTime) : TDateTime
8457: Function DateTimeToFileTime( DateTime : TDateTime) : TFileTime
8458: Function GMTNow : TDateTime
8459: Function GMTToLocalTime( GMT : TDateTime) : TDateTime
8460: Function LocalTimeToGMT( LocalTime : TDateTime) : TDateTime
8461: Procedure XRTLNotImplemented
8462: Procedure XRTLRaiseError( E : Exception)
8463: Procedure XRTLInvalidOperation( ClassName:string; OperationName:string; Description: string)
8464:
8465:

```

```

8466: procedure SIRegister_xrtl_util_Value(CL: TPSPascalCompiler);
8467: begin
8468:   SIRegister_IXRTLValue(CL);
8469:   SIRegister_TXRTLValue(CL);
8470:   //AddTypeS('PXRTLValueArray', '^TXRTLValueArray // will not work
8471:   AddTypeS('TXRTLValueArray', 'array of IXRTLValue
8472: Function XRTLValue( const AValue : Cardinal) : IXRTLValue;
8473: Function XRTLSetValue( const IValue : IXRTLValue; const AValue : Cardinal) : Cardinal;
8474: Function XRTLGetAsCardinal( const IValue : IXRTLValue) : Cardinal
8475: Function XRTLGetAsCardinalDef( const IValue : IXRTLValue; const DefValue : Cardinal) : Cardinal
8476: Function XRTLValue1( const AValue : Integer) : IXRTLValue;
8477: Function XRTLSetValue1( const IValue : IXRTLValue; const AValue : Integer) : Integer;
8478: Function XRTLGetAsInteger( const IValue : IXRTLValue) : Integer
8479: Function XRTLGetAsIntegerDef( const IValue : IXRTLValue; const DefValue : Integer) : Integer
8480: Function XRTLValue2( const AValue : Int64) : IXRTLValue;
8481: Function XRTLSetValue2( const IValue : IXRTLValue; const AValue : Int64) : Int64;
8482: Function XRTLGetAsInt64( const IValue : IXRTLValue) : Int64
8483: Function XRTLGetAsInt64Def( const IValue : IXRTLValue; const DefValue : Int64) : Int64
8484: Function XRTLValue3( const AValue : Single) : IXRTLValue;
8485: Function XRTLSetValue3( const IValue : IXRTLValue; const AValue : Single) : Single;
8486: Function XRTLGetAsSingle( const IValue : IXRTLValue) : Single
8487: Function XRTLGetAsSingleDef( const IValue : IXRTLValue; const DefValue : Single) : Single
8488: Function XRTLValue4( const AValue : Double) : IXRTLValue;
8489: Function XRTLSetValue4( const IValue : IXRTLValue; const AValue : Double) : Double;
8490: Function XRTLGetAsDouble( const IValue : IXRTLValue) : Double
8491: Function XRTLGetAsDoubleDef( const IValue : IXRTLValue; const DefValue : Double) : Double
8492: Function XRTLValue5( const AValue : Extended) : IXRTLValue;
8493: Function XRTLSetValue5( const IValue : IXRTLValue; const AValue : Extended) : Extended;
8494: Function XRTLGetAsExtended( const IValue : IXRTLValue) : Extended
8495: Function XRTLGetAsExtendedDef( const IValue : IXRTLValue; const DefValue : Extended) : Extended
8496: Function XRTLValue6( const AValue : IInterface) : IXRTLValue;
8497: Function XRTLSetValue6( const IValue : IXRTLValue; const AValue : IInterface) : IInterface;
8498: Function XRTLGetAsInterface( const IValue : IXRTLValue) : IInterface;
8499: //Function XRTLGetAsInterface1( const IValue : IXRTLValue; out Obj) : IInterface;
8500: Function XRTLGetAsInterfaceDef( const IValue : IXRTLValue; const DefValue : IInterface) : IInterface;
8501: Function XRTLValue7( const AValue : WideString) : IXRTLValue;
8502: Function XRTLSetValue7( const IValue : IXRTLValue; const AValue : WideString) : WideString;
8503: Function XRTLGetAsWideString( const IValue : IXRTLValue) : WideString
8504: Function XRTLGetAsWideStringDef( const IValue : IXRTLValue; const DefValue : WideString) : WideString
8505: Function XRTLValue8( const AValue : TObject; const AOwnValue : Boolean) : IXRTLValue;
8506: Function XRTLSetValue8( const IValue : IXRTLValue; const AValue : TObject) : TObject;
8507: Function XRTLGetAsObject( const IValue : IXRTLValue; const ADetachOwnership : Boolean) : TObject;
8508: Function XRTLGetAsObjectDef( const IValue:IXRTLValue;const DefValue:TObject;const
ADetachOwnership:Boolean):TObject;
8509: //Function XRTLValue9( const AValue : __Pointer) : IXRTLValue;
8510: //Function XRTLSetValue9( const IValue : IXRTLValue; const AValue : __Pointer) : __Pointer;
8511: //Function XRTLGetAsPointer( const IValue : IXRTLValue) : __Pointer
8512: //Function XRTLGetAsPointerDef( const IValue : IXRTLValue; const DefValue : __Pointer) : __Pointer
8513: Function XRTLValueV( const AValue : Variant) : IXRTLValue;
8514: Function XRTLSetValueV( const IValue : IXRTLValue; const AValue : Variant) : Variant;
8515: Function XRTLGetAsVariant( const IValue : IXRTLValue) : Variant
8516: Function XRTLGetAsVariantDef( const IValue : IXRTLValue; const DefValue : Variant) : Variant
8517: Function XRTLValue10( const AValue : Currency) : IXRTLValue;
8518: Function XRTLSetValue10( const IValue : IXRTLValue; const AValue : Currency) : Currency;
8519: Function XRTLGetAsCurrency( const IValue : IXRTLValue) : Currency
8520: Function XRTLGetAsCurrencyDef( const IValue : IXRTLValue; const DefValue : Currency) : Currency
8521: Function XRTLValue11( const AValue : Comp) : IXRTLValue;
8522: Function XRTLSetValue11( const IValue : IXRTLValue; const AValue : Comp) : Comp;
8523: Function XRTLGetAsComp( const IValue : IXRTLValue) : Comp
8524: Function XRTLGetAsCompDef( const IValue : IXRTLValue; const DefValue : Comp) : Comp
8525: Function XRTLValue12( const AValue : TClass) : IXRTLValue;
8526: Function XRTLSetValue12( const IValue : IXRTLValue; const AValue : TClass) : TClass;
8527: Function XRTLGetAsClass( const IValue : IXRTLValue) : TClass
8528: Function XRTLGetAsClassDef( const IValue : IXRTLValue; const DefValue : TClass) : TClass
8529: Function XRTLValue13( const AValue : TGUID) : IXRTLValue;
8530: Function XRTLSetValue13( const IValue : IXRTLValue; const AValue : TGUID) : TGUID;
8531: Function XRTLGetAsGUID( const IValue : IXRTLValue) : TGUID
8532: Function XRTLGetAsGUIDDef( const IValue : IXRTLValue; const DefValue : TGUID) : TGUID
8533: Function XRTLValue14( const AValue : Boolean) : IXRTLValue;
8534: Function XRTLSetValue14( const IValue : IXRTLValue; const AValue : Boolean) : Boolean;
8535: Function XRTLGetAsBoolean( const IValue : IXRTLValue) : Boolean
8536: Function XRTLGetAsBooleanDef( const IValue : IXRTLValue; const DefValue : Boolean) : Boolean
8537: end;
8538:
8539: //*****unit uPSI_GR32;*****
8540:
8541: Function Color32( WinColor : TColor) : TColor32;
8542: Function Color321( R, G, B : Byte; A : Byte) : TColor32;
8543: Function Color322( Index : Byte; var Palette : TPalette32) : TColor32;
8544: Function Gray32( Intensity : Byte; Alpha : Byte) : TColor32
8545: Function WinColor( Color32 : TColor32) : TColor
8546: Function ArrayOfColor32( Colors : array of TColor32) : TArrayOfColor32
8547: Procedure Color32ToRGB( Color32 : TColor32; var R, G, B : Byte)
8548: Procedure Color32ToRGBA( Color32 : TColor32; var R, G, B, A : Byte)
8549: Function Color32Components( R, G, B, A : Boolean) : TColor32Components
8550: Function RedComponent( Color32 : TColor32) : Integer
8551: Function GreenComponent( Color32 : TColor32) : Integer
8552: Function BlueComponent( Color32 : TColor32) : Integer
8553: Function AlphaComponent( Color32 : TColor32) : Integer

```

```

8554: Function Intensity( Color32 : TColor32 ) : Integer
8555: Function SetAlpha( Color32 : TColor32; NewAlpha : Integer ) : TColor32
8556: Function HSLtoRGB( H, S, L : Single ) : TColor32;
8557: Procedure RGBtoHSL( RGB : TColor32; out H, S, L : Single);
8558: Function HSLtoRGB1( H, S, L : Integer ) : TColor32;
8559: Procedure RGBtoHSL1( RGB : TColor32; out H, S, L : Byte);
8560: Function WinPalette( const P : TPalette32 ) : HPALETTE
8561: Function FloatPoint( X, Y : Single ) : TFloatPoint;
8562: Function FloatPoint1( const P : TPoint ) : TFloatPoint;
8563: Function FloatPoint2( const FXP : TFixedPoint ) : TFloatPoint;
8564: Function FixedPoint( X, Y : Integer ) : TFixedPoint;
8565: Function FixedPoint1( X, Y : Single ) : TFixedPoint;
8566: Function FixedPoint2( const P : TPoint ) : TFixedPoint;
8567: Function FixedPoint3( const FP : TFloatPoint ) : TFixedPoint;
8568: AddTypes( 'TRectRounding', '( rrClosest, rrOutside, rrInside )
8569: Function MakeRect( const L, T, R, B : Integer ) : TRect;
8570: Function MakeRect1( const FR : TFloatRect; Rounding : TRectRounding ) : TRect;
8571: Function MakeRect2( const FXR : TRect; Rounding : TRectRounding ) : TRect;
8572: Function GFixedRect( const L, T, R, B : TFixed ) : TRect;
8573: Function FixedRect1( const ARect : TRect ) : TRect;
8574: Function FixedRect2( const FR : TFloatRect ) : TRect;
8575: Function GFloatRect( const L, T, R, B : TFloat ) : TFloatRect;
8576: Function FloatRect1( const ARect : TRect ) : TFloatRect;
8577: Function FloatRect2( const FXR : TRect ) : TFloatRect;
8578: Function GIntersectRect( out Dst : TRect; const R1, R2 : TRect ) : Boolean;
8579: Function IntersectRect1( out Dst : TFloatRect; const FR1, FR2 : TFloatRect ) : Boolean;
8580: Function GUnionRect( out Rect : TRect; const R1, R2 : TRect ) : Boolean;
8581: Function UnionRect1( out Rect : TFloatRect; const R1, R2 : TFloatRect ) : Boolean;
8582: Function GEqualRect( const R1, R2 : TRect ) : Boolean;
8583: Function EqualRect1( const R1, R2 : TFloatRect ) : Boolean;
8584: Procedure GInflateRect( var R : TRect; Dx, Dy : Integer);
8585: Procedure InflateRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8586: Procedure GOffsetRect( var R : TRect; Dx, Dy : Integer);
8587: Procedure OffsetRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8588: Function IsRectEmpty( const R : TRect ) : Boolean;
8589: Function IsRectEmpty1( const FR : TFloatRect ) : Boolean;
8590: Function GPtInRect( const R : TRect; const P : TPoint ) : Boolean;
8591: Function PtInRect1( const R : TFloatRect; const P : TPoint ) : Boolean;
8592: Function PtInRect2( const R : TRect; const P : TFloatPoint ) : Boolean;
8593: Function PtInRect3( const R : TFloatRect; const P : TFloatPoint ) : Boolean;
8594: Function EqualRectSize( const R1, R2 : TRect ) : Boolean;
8595: Function EqualRectSize1( const R1, R2 : TFloatRect ) : Boolean;
8596: Function MessageBeep( uType : UINT ) : BOOL
8597: Function ShowCursor( bShow : BOOL ) : Integer
8598: Function SetCursorPos( X, Y : Integer ) : BOOL
8599: Function SetCursor( hCursor : HICON ) : HCURSOR
8600: Function GetCursorPos( var lpPoint : TPoint ) : BOOL
8601: //Function ClipCursor( lpRect : PRect ) : BOOL
8602: Function GetClipCursor( var lpRect : TRect ) : BOOL
8603: Function GetCursor : HCURSOR
8604: Function CreateCaret( hWnd : HWND; hBitmap : HBITMAP; nWidth, nHeight : Integer ) : BOOL
8605: Function GetCaretBlinkTime : UINT
8606: Function SetCaretBlinkTime( uMSeconds : UINT ) : BOOL
8607: Function DestroyCaret : BOOL
8608: Function HideCaret( hWnd : HWND ) : BOOL
8609: Function ShowCaret( hWnd : HWND ) : BOOL
8610: Function SetCaretPos( X, Y : Integer ) : BOOL
8611: Function GetCaretPos( var lpPoint : TPoint ) : BOOL
8612: Function ClientToScreen( hWnd : HWND; var lpPoint : TPoint ) : BOOL
8613: Function ScreenToClient( hWnd : HWND; var lpPoint : TPoint ) : BOOL
8614: Function MapWindowPoints( hWndFrom, hWndTo:HWND; var lpPoints, cPoints : UINT ) : Integer
8615: Function WindowFromPoint( Point : TPoint ) : HWND
8616: Function ChildWindowFromPoint( hWndParent : HWND; Point : TPoint ) : HWND
8617:
8618:
8619: procedure SIRegister_GR32_Math(CL: TPSPascalCompiler);
8620: begin
8621: Function FixedFloor( A : TFixed ) : Integer
8622: Function FixedCeil( A : TFixed ) : Integer
8623: Function FixedMul( A, B : TFixed ) : TFixed
8624: Function FixedDiv( A, B : TFixed ) : TFixed
8625: Function OneOver( Value : TFixed ) : TFixed
8626: Function FixedRound( A : TFixed ) : Integer
8627: Function FixedSqr( Value : TFixed ) : TFixed
8628: Function FixedSqrtLP( Value : TFixed ) : TFixed
8629: Function FixedSqrthp( Value : TFixed ) : TFixed
8630: Function FixedCombine( W, X, Y : TFixed ) : TFixed
8631: Procedure GRSinCos( const Theta : TFloat; out Sin, Cos : TFloat);
8632: Procedure GRSinCos1( const Theta, Radius : Single; out Sin, Cos : Single);
8633: Function GRHypot( const X, Y : TFloat ) : TFloat;
8634: Function Hypot1( const X, Y : Integer ) : Integer;
8635: Function FastSqrt( const Value : TFloat ) : TFloat
8636: Function FastSqrtBab1( const Value : TFloat ) : TFloat
8637: Function FastSqrtBab2( const Value : TFloat ) : TFloat
8638: Function FastInvSqrt( const Value : Single ) : Single;
8639: Function MulDiv( Multiplicand, Multiplier, Divisor : Integer ) : Integer
8640: Function GRIsPowerOf2( Value : Integer ) : Boolean
8641: Function PrevPowerOf2( Value : Integer ) : Integer
8642: Function NextPowerOf2( Value : Integer ) : Integer

```



```

8643: Function Average( A, B : Integer) : Integer
8644: Function GRSign( Value : Integer) : Integer
8645: Function FloatMod( x, y : Double) : Double
8646: end;
8647:
8648: procedure SIRegister_GR32_LowLevel(CL: TPSPascalCompiler);
8649: begin
8650:   Function Clamp( const Value : Integer) : Integer;
8651:   Procedure GRFillWord( var X, Count : Cardinal; Value : Longword)
8652:   Function StackAlloc( Size : Integer) : Pointer
8653:   Procedure StackFree( P : Pointer)
8654:   Procedure Swap( var A, B : Pointer);
8655:   Procedure Swap1( var A, B : Integer);
8656:   Procedure Swap2( var A, B : TFixed);
8657:   Procedure Swap3( var A, B : TColor32);
8658:   Procedure TestSwap( var A, B : Integer);
8659:   Procedure TestSwap1( var A, B : TFixed);
8660:   Function TestClip( var A, B : Integer; const Size : Integer) : Boolean;
8661:   Function TestClip1( var A, B : Integer; const Start, Stop : Integer) : Boolean;
8662:   Function GRConstrain( const Value, Lo, Hi : Integer) : Integer;
8663:   Function Constrain1( const Value, Lo, Hi : Single) : Single;
8664:   Function SwapConstrain( const Value:Integer; Constrain1,Constrain2:Integer) : Integer
8665:   Function GRMin( const A, B, C : Integer) : Integer;
8666:   Function GRMax( const A, B, C : Integer) : Integer;
8667:   Function Clamp( Value, Max : Integer) : Integer;
8668:   Function Clamp1( Value, Min, Max : Integer) : Integer;
8669:   Function Wrap( Value, Max : Integer) : Integer;
8670:   Function Wrap1( Value, Min, Max : Integer) : Integer;
8671:   Function Wrap3( Value, Max : Single) : Single;;
8672:   Function WrapPow2( Value, Max : Integer) : Integer;
8673:   Function WrapPow21( Value, Min, Max : Integer) : Integer;
8674:   Function Mirror( Value, Max : Integer) : Integer;
8675:   Function Mirror1( Value, Min, Max : Integer) : Integer;
8676:   Function MirrorPow2( Value, Max : Integer) : Integer;
8677:   Function MirrorPow21( Value, Min, Max : Integer) : Integer;
8678:   Function GetOptimalWrap( Max : Integer) : TWrapProc;
8679:   Function GetOptimalWrap1( Min, Max : Integer) : TWrapProcEx;
8680:   Function GetOptimalMirror( Max : Integer) : TWrapProc;
8681:   Function GetOptimalMirror1( Min, Max : Integer) : TWrapProcEx;
8682:   Function GetWrapProc( WrapMode : TWrapMode) : TWrapProc;
8683:   Function GetWrapProc1( WrapMode : TWrapMode; Max : Integer) : TWrapProc;
8684:   Function GetWrapProcEx( WrapMode : TWrapMode) : TWrapProcEx;
8685:   Function GetWrapProcEx1( WrapMode : TWrapMode; Min, Max : Integer):TWrapProcEx;
8686:   Function Div255( Value : Cardinal) : Cardinal
8687:   Function SAR_4( Value : Integer) : Integer
8688:   Function SAR_8( Value : Integer) : Integer
8689:   Function SAR_9( Value : Integer) : Integer
8690:   Function SAR_11( Value : Integer) : Integer
8691:   Function SAR_12( Value : Integer) : Integer
8692:   Function SAR_13( Value : Integer) : Integer
8693:   Function SAR_14( Value : Integer) : Integer
8694:   Function SAR_15( Value : Integer) : Integer
8695:   Function SAR_16( Value : Integer) : Integer
8696:   Function ColorSwap( WinColor : TColor) : TColor32
8697: end;
8698:
8699: procedure SIRegister_GR32_Filters(CL: TPSPascalCompiler);
8700: begin
8701:   AddTypeS('TLogicalOperator', '( loXOR, loAND, loOR )
8702:   Procedure CopyComponents( Dst, Src : TCustomBitmap32; Components : TColor32Components);
8703:   Procedure CopyComponents1(Dst:TCustomBmap32;DstX,
      DstY:Int;Src:TCustomBmap32;SrcRect:TRect;Components:TColor32Comp;
8704:   Procedure AlphaToGrayscale( Dst, Src : TCustomBitmap32)
8705:   Procedure ColorToGrayscale( Dst, Src : TCustomBitmap32; PreserveAlpha : Boolean)
8706:   Procedure IntensityToAlpha( Dst, Src : TCustomBitmap32)
8707:   Procedure Invert( Dst, Src : TCustomBitmap32; Components : TColor32Components)
8708:   Procedure InvertRGB( Dst, Src : TCustomBitmap32)
8709:   Procedure ApplyLUT( Dst, Src : TCustomBitmap32; const LUT : TLUT8; PreserveAlpha : Boolean)
8710:   Procedure ChromaKey( ABitmap : TCustomBitmap32; TrColor : TColor32)
8711:   Function CreateBitmap( Components : TColor32Components) : TColor32
8712:   Procedure ApplyBitmap(Dst:TCustomBitmap32; DstX,DstY:Integer; Src:TCustomBitmap32; SrcRect : TRect;
      Bitmask : TColor32; LogicalOperator : TLogicalOperator);
8713:   Procedure
      ApplyBitmap1(ABitmap:TCustomBitmap32;ARect:TRect;Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8714:   Procedure CheckParams( Dst, Src : TCustomBitmap32; ResizeDst : Boolean)
8715: end;
8716:
8717:
8718: procedure SIRegister_JclNTFS(CL: TPSPascalCompiler);
8719: begin
8720:   AddClassN(FindClass('TOBJECT'),'EJclNtfsError
8721:   AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )
8722:   Function NtfsGetCompression( const FileName : string; var State : Short) : Boolean;
8723:   Function NtfsGetCompression1( const FileName : string) : TFileCompressionState;
8724:   Function NtfsSetCompression( const FileName : string; const State : Short) : Boolean
8725:   Procedure NtfsSetFileCompression( const FileName : string; const State : TFileCompressionState)
8726:   Procedure NtfsSetDirectoryTreeCompression(const Directory: string; const State : TFileCompressionState)
8727:   Procedure NtfsSetDefaultFileCompression(const Directory: string; const State:TFileCompressionState)
8728:   Procedure NtfsSetPathCompression(const Path:string;const State:TFileCompressionState;Recursive:Boolean;

```

```

8729: //AddTypeS('TNtfsAllocRanges', 'record Entries : Integer; Data : PFileAlloca'
8730: //+'tedRangeBuffer; MoreData : Boolean; end
8731: Function NtfsSetSparse( const FileName : string ) : Boolean
8732: Function NtfsZeroDataByHandle( const Handle : THandle; const First, Last : Int64 ) : Boolean
8733: Function NtfsZeroDataByName( const FileName : string; const First, Last : Int64 ) : Boolean
8734: //Function NtfsQueryAllocRanges(const FileName:string;Offset,Count:Int64;var
      Ranges:TNtfsAllocRanges):Boolean;
8735: //Function NtfsGetAllocRangeEntry( const Ranges : TNtfsAllocRanges;
      Index:Integer):TFileAllocatedRangeBuffer
8736: Function NtfsSparseStreamsSupported( const Volume : string ) : Boolean
8737: Function NtfsGetSparse( const FileName : string ) : Boolean
8738: Function NtfsDeleteReparsePoint( const FileName : string; ReparseTag : DWORD ) : Boolean
8739: Function NtfsSetReparsePoint( const FileName : string; var ReparseData, Size : Longword ) : Boolean
8740: //Function NtfsGetReparsePoint(const FileName:string; var ReparseData:TReparseGuidDataBuffer):Boolean
8741: Function NtfsGetReparseTag( const Path : string; var Tag : DWORD ) : Boolean
8742: Function NtfsReparsePointsSupported( const Volume : string ) : Boolean
8743: Function NtfsFileHasReparsePoint( const Path : string ) : Boolean
8744: Function NtfsFolderMountPoint( const Path : string ) : Boolean
8745: Function NtfsMountDeviceAsDrive( const Device : string; Drive : Char ) : Boolean
8746: Function NtfsMountVolume( const Volume : Char; const MountPoint : string ) : Boolean
8747: AddTypeS('TTopLock', '( olExclusive, olReadOnly, olBatch, olFilter )
8748: Function NtfsOpLockAckClosePending( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8749: Function NtfsOpLockBreakAckNo2( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8750: Function NtfsOpLockBreakAcknowledge( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8751: Function NtfsOpLockBreakNotify( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8752: Function NtfsRequestOpLock( Handle : THandle; Kind : TTopLock; Overlapped : TOverlapped ) : Boolean
8753: Function NtfsCreateJunctionPoint( const Source, Destination : string ) : Boolean
8754: Function NtfsDeleteJunctionPoint( const Source : string ) : Boolean
8755: Function NtfsGetJunctionPointDestination( const Source : string; var Destination : string ) : Boolean
8756: AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
      +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile )
8757: AddTypeS('TStreamIds', 'set of TStreamId
8758: AddTypeS('TInternalFindStreamData', 'record FileHandle : THandle; Context '
      +': __Pointer; StreamIds : TStreamIds; end
8759: AddTypeS('TFindStreamData', 'record internal : TInternalFindStreamData; At'
      +':tributes : DWORD; StreamID : TStreamId; Name : WideString; Size : Int64; end
8760: Function NtfsFindFirstStream(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Boolean;
8761: Function NtfsFindNextStream( var Data : TFindStreamData ) : Boolean
8762: Function NtfsFindStreamClose( var Data : TFindStreamData ) : Boolean
8763: Function NtfsCreateHardLink( const LinkFileName, ExistingFileName : string ) : Boolean
8764: AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end
8765: Function NtfsGetHardLinkInfo( const FileName : string; var Info : TNtfsHardLinkInfo ) : Boolean
8766: Function NtfsFindHardLinks(const Path:string;const FileIndexHigh,FileIndexLow:Cardinal;const
      List:TStrings):Bool;
8770: Function NtfsDeleteHardLinks( const FileName : string ) : Boolean
8771: Function JclAppInstances : TJclAppInstances;
8772: Function JclAppInstances1( const UniqueAppIdGuidStr : string ) : TJclAppInstances;
8773: Function ReadMessageCheck( var Message: TMessage;const IgnoredOriginatorWnd: HWND ) : TJclAppInstDataKind
8774: Procedure ReadMessageData( const Message : TMessage; var Data : __Pointer; var Size : Integer )
8775: Procedure ReadMessageString( const Message : TMessage; var S : string )
8776: Procedure ReadMessageStrings( const Message : TMessage; const Strings : TStrings )
8777:
8778:
8779: (*-----*)
8780: procedure SIRegister_JclGraphics(CL: TPSPascalCompiler);
8781: begin
8782:   FindClass('TOBJECT'),'EJclGraphicsError
8783:   TDynDynIntegerArrayArray', 'array of TDynIntegerArray
8784:   TDynPointArray', 'array of TPoint
8785:   TDynDynPointArrayArray', 'array of TDynPointArray
8786:   TPointF', 'record X : Single; Y : Single; end
8787:   TDynPointArrayF', 'array of TPointF
8788:   TDrawMode2', '( dmOpaque, dmBlend )
8789:   TStretchFilter2', '( sfNearest, sfLinear, sfSpline )
8790:   TConversionKind', '( ckRed, ckGreen, ckBlue, ckAlpha, ckUniformRGB, ckWeightedRGB )
8791:   TResamplingFilter', '( rfBox, rfTriangle, rfHermite, rfBell, rfSpline, rfLanczos3, rfMitchell )
8792:   TMatrix3d', 'record array[0..2,0..2] of extended end
8793:   TDynDynPointArrayArrayF', 'array of TDynPointArrayF
8794:   TScanLine', 'array of Integer
8795:   TScanLines', 'array of TScanLine
8796:   TColorChannel', '( ccRed, ccGreen, ccBlue, ccAlpha )
8797:   TGradientDirection', '( gdVertical, gdHorizontal )
8798:   TPolyFillMode', '( fmAlternate, fmWinding )
8799:   TJclRegionCombineOperator', '( coAnd, coDiff, coOr, coXor )
8800:   TJclRegionBitmapMode', '( rmInclude, rmExclude )
8801:   TJclRegionKind', '( rkNull, rkSimple, rkComplex, rkError )
8802:   SIRegister_TJclDesktopCanvas(CL);
8803:   FindClass('TOBJECT'),'TJclRegion
8804:   SIRegister_TJclRegionInfo(CL);
8805:   SIRegister_TJclRegion(CL);
8806:   SIRegister_TJclThreadPersistent(CL);
8807:   SIRegister_TJclCustomMap(CL);
8808:   SIRegister_TJclBitmap32(CL);
8809:   SIRegister_TJclByteMap(CL);
8810:   SIRegister_TJclTransformation(CL);
8811:   SIRegister_TJclLinearTransformation(CL);
8812:   Procedure Stretch(NewWidth,
      NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Source:TGraphic;Target:TBitmap);
8813:   Procedure Stretch1(NewWidth,NewHeight:Cardinal;Filter:TResamplingFilter;Radius:Single;Bitmap:TBitmap);

```

```

8814: Procedure DrawBitmap( DC : HDC; Bitmap : HBitmap; X, Y, Width, Height : Integer)
8815: Function GetAntialiasedBitmap( const Bitmap : TBitmap) : TBitmap
8816: Procedure BitmapToJpeg( const FileName : string)
8817: Procedure JpegToBitmap( const FileName : string)
8818: Function ExtractIconCount( const FileName : string) : Integer
8819: Function BitmapToIconJ( Bitmap : HBITMAP; cx, cy : Integer) : HICON
8820: Function IconToBitmapJ( Icon : HICON) : HBITMAP
8821: Procedure
BlockTransfer(Dst:TJclBitmap32;DstX:Int;DstY:Int;Src:TJclBitmap32;SrcRect:TRect;CombineOp:TDrawMode)
8822: Procedure StretchTransfer(Dst:TJclBitmap32;
DstRect:TRect;Src:TJclBitmap32;SrcRect:TRect;StretchFilter:TStretchFilter; CombineOp : TDrawMode)
8823: Procedure Transform( Dst, Src : TJclBitmap32; SrcRect : TRect; Transformation : TJclTransformation)
8824: Procedure SetBorderTransparent( ABitmap : TJclBitmap32; ARect : TRect)
8825: Function FillGradient(DC:HDC; ARect:TRect; ColorCount:Integer; StartColor,EndColor:TColor;ADirection :
TGradientDirection) : Boolean;
8826: Function CreateRegionFromBitmap(Bitmap: TBitmap; RegionColor:TColor;
RegionBitmapMode:TJclRegionBitmapMode): HRGN
8827: Procedure Screenshot( bm : TBitmap; Left, Top, Width, Height : Integer; Window : HWND);
8828: Procedure Screenshot1( bm : TBitmap);
8829: Procedure PolyLineTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8830: Procedure PolyLineAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8831: Procedure PolyLineFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8832: Procedure PolygonTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8833: Procedure PolygonAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8834: Procedure PolygonFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8835: Procedure PolyPolygonTS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8836: Procedure PolyPolygonAS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8837: Procedure PolyPolygonFS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArrayF;Color:TColor32);
8838: Procedure AlphaToGrayscale( Dst, Src : TJclBitmap32)
8839: Procedure IntensityToAlpha( Dst, Src : TJclBitmap32)
8840: Procedure Invert( Dst, Src : TJclBitmap32)
8841: Procedure InvertRGB( Dst, Src : TJclBitmap32)
8842: Procedure ColorToGrayscale( Dst, Src : TJclBitmap32)
8843: Procedure ApplyLUT( Dst, Src : TJclBitmap32; const LUT : TLUT8)
8844: Procedure SetGamma( Gamma : Single)
8845: end;
8846:
8847: (*-----*)
8848: procedure SIRegister_JclSynch(CL: TPSPascalCompiler);
8849: begin
8850: Function LockedAdd( var Target : Integer; Value : Integer) : Integer
8851: Function LockedCompareExchange( var Target : Integer; Exch, Comp : Integer) : Integer;
8852: Function LockedCompareExchange1( var Target : __Pointer; Exch, Comp : __Pointer) : Pointer;
8853: Function LockedDec( var Target : Integer) : Integer
8854: Function LockedExchange( var Target : Integer; Value : Integer) : Integer
8855: Function LockedExchangeAdd( var Target : Integer; Value : Integer) : Integer
8856: Function LockedExchangeDec( var Target : Integer) : Integer
8857: Function LockedExchangeInc( var Target : Integer) : Integer
8858: Function LockedExchangeSub( var Target : Integer; Value : Integer) : Integer
8859: Function LockedInc( var Target : Integer) : Integer
8860: Function LockedSub( var Target : Integer; Value : Integer) : Integer
8861: TJclWaitResult', '( wrAbandoned, wrError, wrIoCompletion, wrSignaled, wrTimeout )
8862: SIRegister_TJclDispatcherObject(CL);
8863: Function WaitForMultipleObjects(const Objects:array of
TJclDispatcherObject;WaitAll:Bool;Timeout:Cardinal):Cardinal;
8864: Function WaitAlertableForMultipleObjects(const Objects : array of TJclDispatcherObject; WaitAll:Bool;
TimeOut : Cardinal):Cardinal
8865: SIRegister_TJclCriticalSection(CL);
8866: SIRegister_TJclCriticalSectionEx(CL);
8867: SIRegister_TJclEvent(CL);
8868: SIRegister_TJclWaitableTimer(CL);
8869: SIRegister_TJclSemaphore(CL);
8870: SIRegister_TJclMutex(CL);
8871: POptexSharedInfo', '^TOptexSharedInfo // will not work
8872: TOptexSharedInfo', 'record SpinCount: Int; LockCount: Int; ThreadId: Longword; RecursionCount: Int; end
8873: SIRegister_TJclOptex(CL);
8874: TMrewPreferred', '( mpReaders, mpWriters, mpEqual )
8875: TMrewThreadInfo', 'record ThreadId : Longword; RecursionCount: Integer; Reader : Boolean; end
8876: TMrewThreadInfoArray', 'array of TMrewThreadInfo
8877: SIRegister_TJclMultiReadExclusiveWrite(CL);
8878: PMetSectSharedInfo', '^TMetSectSharedInfo // will not work
8879: TMetSectSharedInfo', 'record Initialized : LongBool; SpinLock : '
+ 'Longint; ThreadsWaiting : Longint; AvailableCount : Longint; MaximumCount : Longint; end
8880: PMeteredSection', '^TMeteredSection // will not work
8881: TMeteredSection', 'record Event : THandle; FileMap : THandle; SharedInfo : PMetSectSharedInfo; end
8882: SIRegister_TJclMeteredSection(CL);
8883: TEventInfo', 'record EventType : Longint; Signaled : LongBool; end
8884: TMutexInfo', 'record SignalState : Longint; Owned : Boolean; Abandoned : Boolean; end
8885: TSemaphoreCounts', 'record CurrentCount : Longint; MaximumCount : Longint; end
8886: TTimerInfo', 'record Remaining : TLargeInteger; Signaled : LongBool; end
8887: Function QueryCriticalSection( CS : TJclCriticalSection; var Info : TRTLCriticalSection) : Boolean
8888: Function QueryEvent( Handle : THandle; var Info : TEventInfo) : Boolean
8889: Function QueryMutex( Handle : THandle; var Info : TMutexInfo) : Boolean
8890: Function QuerySemaphore( Handle : THandle; var Info : TSemaphoreCounts) : Boolean
8891: Function QueryTimer( Handle : THandle; var Info : TTimerInfo) : Boolean
8892: FindClass('TOBJECT'),'EJclWin32HandleObjectError
8893: FindClass('TOBJECT'),'EJclDispatcherObjectError
8894: FindClass('TOBJECT'),'EJclCriticalSectionError
8895: FindClass('TOBJECT'),'EJclEventError
8896: FindClass('TOBJECT'),'EJclEventError

```

```

8897: FindClass('TOBJECT'),'EJclWaitableTimerError
8898: FindClass('TOBJECT'),'EJclSemaphoreError
8899: FindClass('TOBJECT'),'EJclMutexError
8900: FindClass('TOBJECT'),'EJclMeteredSectionError
8901: end;
8902:
8903:
8904: //*****unit uPSI_mORMotReport;
8905: Procedure SetCurrentPrinterAsDefault
8906: Function CurrentPrinterName : string
8907: Function mCurrentPrinterPaperSize : string
8908: Procedure UseDefaultPrinter
8909:
8910: procedure SIRegisterTSTREAM(C1: TPSPascalCompiler);
8911: begin
8912:   with FindClass('TOBJECT'), 'TStream' do begin
8913:     IsAbstract := True;
8914:     //RegisterMethod('Function Read( var Buffer, Count : Longint) : Longint
8915:     // RegisterMethod('Function Write( const Buffer, Count : Longint) : Longint
8916:     function Read(Buffer:String;Count:LongInt):LongInt
8917:     function Write(Buffer:String;Count:LongInt):LongInt
8918:     function ReadString(Buffer:String;Count:LongInt):LongInt //FileStream
8919:     function WriteString(Buffer:String;Count:LongInt):LongInt
8920:     function ReadInt(Buffer:integer;Count:LongInt):LongInt
8921:     function WriteInt(Buffer:integer;Count:LongInt):LongInt
8922:     procedure ReadAB(Buffer: TByteArray;Count:LongInt)
8923:     procedure WriteAB(Buffer: TByteArray;Count:LongInt)
8924:     procedure ReadABD(Buffer: TByteDynArray;Count:LongInt)
8925:     procedure WriteABD(Buffer: TByteDynArray;Count:LongInt)
8926:     procedure ReadAC(Buffer: TCharArray;Count:LongInt)
8927:     procedure WriteAC(Buffer: TCharArray;Count:LongInt)
8928:     procedure ReadACD(Buffer: TCharDynArray;Count:LongInt)
8929:     procedure WriteACD(Buffer: TCharDynArray;Count:LongInt)
8930:
8931:     function Seek(Offset:LongInt;Origin:Word):LongInt
8932:     procedure ReadBuffer(Buffer:String;Count:LongInt)
8933:     procedure WriteBuffer(Buffer:String;Count:LongInt)
8934:     procedure ReadBufferInt(Buffer:Integer;Count:LongInt)';
8935:     procedure WriteBufferInt(Buffer:Integer;Count:LongInt)';
8936:     procedure ReadBufferFloat(Buffer:Double;Count:LongInt)';
8937:     Procedure WriteBufferFloat(Buffer:Double;Count:LongInt)';
8938:
8939:     procedure ReadBufferAB(Buffer: TByteArray;Count:LongInt)
8940:     procedure WriteBufferAB(Buffer: TByteArray;Count:LongInt)
8941:     procedure ReadBufferABD(Buffer: TByteDynArray;Count:LongInt)
8942:     procedure WriteBufferABD(Buffer: TByteDynArray;Count:LongInt)
8943:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
8944:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
8945:     procedure ReadBufferAWD(Buffer: TWordDynArray;Count:LongInt)
8946:     procedure WriteBufferAWD(Buffer: TWordDynArray;Count:LongInt)
8947:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
8948:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
8949:     procedure ReadBufferAC(Buffer: TCharArray;Count:LongInt)
8950:     procedure WriteBufferAC(Buffer: TCharArray;Count:LongInt)
8951:     procedure ReadBufferACD(Buffer: TCharDynArray;Count:LongInt)
8952:     procedure WriteBufferACD(Buffer: TCharDynArray;Count:LongInt)
8953:
8954:     procedure ReadBufferP(Buffer: PChar;Count:LongInt)
8955:     procedure WriteBufferP(Buffer: PChar;Count:LongInt)
8956:     procedure ReadBufferO(Buffer: TObject;Count:LongInt)';
8957:     procedure WriteBufferO(Buffer: TObject;Count:LongInt)';
8958:     //READBUFFERAC
8959:     function InstanceSize: Longint
8960:     Procedure FixupResourceHeader( FixupInfo : Integer)
8961:     Procedure ReadResHeader
8962:
8963:     {$IFDEF DELPHI4UP}
8964:     function CopyFrom(Source:TStream;Count:Int64):LongInt
8965:     {$ELSE}
8966:     function CopyFrom(Source:TStream;Count:Integer):LongInt
8967:     {$ENDIF}
8968:     RegisterProperty('Position', 'LongInt', iptrw);
8969:     RegisterProperty('Size', 'LongInt', iptrw);
8970:   end;
8971: end;
8972:
8973:
8974: { *****
8975:   Unit DMATH - Interface for DMATH.DLL
8976:   ***** }
8977: // see more docs/dmath_manual.pdf
8978:
8979: Function InitEval : Integer
8980: Procedure SetVariable( VarName : Char; Value : Float)
8981: Procedure SetFunction( FuncName : String; Wrapper : TWrapper)
8982: Function Eval( ExpressionString : String) : Float
8983:
8984: unit dmath; //types are in built, others are external in DLL
8985: interface

```



```

8986: {$IFDEF DELPHI}
8987: uses
8988:   StdCtrls, Graphics;
8989: {$ENDIF}
8990: { -----
8991:   Types and constants
8992: ----- }
8993: {$i types.inc}
8994: { -----
8995:   Error handling
8996: ----- }
8997: procedure SetErrCode(ErrCode : Integer); external 'dmath';
8998: { Sets the error code }
8999: function DefaultVal(ErrCode : Integer; DefVal : Float) : Float; external 'dmath';
9000: { Sets error code and default function value }
9001: function MathErr : Integer; external 'dmath';
9002: { Returns the error code }
9003: { -----
9004:   Dynamic arrays
9005: ----- }
9006: procedure SetAutoInit(AutoInit : Boolean); external 'dmath';
9007: { Sets the auto-initialization of arrays }
9008: procedure DimVector(var V : TVector; Ub : Integer); external 'dmath';
9009: { Creates floating point vector V[0..Ub] }
9010: procedure DimIntVector(var V : TIntVector; Ub : Integer); external 'dmath';
9011: { Creates integer vector V[0..Ub] }
9012: procedure DimCompVector(var V : TCompVector; Ub : Integer); external 'dmath';
9013: { Creates complex vector V[0..Ub] }
9014: procedure DimBoolVector(var V : TBoolVector; Ub : Integer); external 'dmath';
9015: { Creates boolean vector V[0..Ub] }
9016: procedure DimStrVector(var V : TStrVector; Ub : Integer); external 'dmath';
9017: { Creates string vector V[0..Ub] }
9018: procedure DimMatrix(var A : TMatrix; Ub1, Ub2 : Integer); external 'dmath';
9019: { Creates floating point matrix A[0..Ub1, 0..Ub2] }
9020: procedure DimIntMatrix(var A : TIntMatrix; Ub1, Ub2 : Integer); external 'dmath';
9021: { Creates integer matrix A[0..Ub1, 0..Ub2] }
9022: procedure DimCompMatrix(var A : TCompMatrix; Ub1, Ub2 : Integer); external 'dmath';
9023: { Creates complex matrix A[0..Ub1, 0..Ub2] }
9024: procedure DimBoolMatrix(var A : TBoolMatrix; Ub1, Ub2 : Integer); external 'dmath';
9025: { Creates boolean matrix A[0..Ub1, 0..Ub2] }
9026: procedure DimStrMatrix(var A : TStrMatrix; Ub1, Ub2 : Integer); external 'dmath';
9027: { Creates string matrix A[0..Ub1, 0..Ub2] }
9028: { -----
9029:   Minimum, maximum, sign and exchange
9030: ----- }
9031: function FMin(X, Y : Float) : Float; external 'dmath';
9032: { Minimum of 2 reals }
9033: function FMax(X, Y : Float) : Float; external 'dmath';
9034: { Maximum of 2 reals }
9035: function IMin(X, Y : Integer) : Integer; external 'dmath';
9036: { Minimum of 2 integers }
9037: function IMax(X, Y : Integer) : Integer; external 'dmath';
9038: { Maximum of 2 integers }
9039: function Sgn(X : Float) : Integer; external 'dmath';
9040: { Sign (returns 1 if X = 0) }
9041: function Sgn0(X : Float) : Integer; external 'dmath';
9042: { Sign (returns 0 if X = 0) }
9043: function DSgn(A, B : Float) : Float; external 'dmath';
9044: { Sgn(B) * |A| }
9045: procedure FSwap(var X, Y : Float); external 'dmath';
9046: { Exchange 2 reals }
9047: procedure ISwap(var X, Y : Integer); external 'dmath';
9048: { Exchange 2 integers }
9049: { -----
9050:   Rounding functions
9051: ----- }
9052: function RoundN(X : Float; N : Integer) : Float; external 'dmath';
9053: { Rounds X to N decimal places }
9054: function Ceil(X : Float) : Integer; external 'dmath';
9055: { Ceiling function }
9056: function Floor(X : Float) : Integer; external 'dmath';
9057: { Floor function }
9058: { -----
9059:   Logarithms, exponentials and power
9060: ----- }
9061: function Expo(X : Float) : Float; external 'dmath';
9062: { Exponential }
9063: function Exp2(X : Float) : Float; external 'dmath';
9064: { 2^X }
9065: function Exp10(X : Float) : Float; external 'dmath';
9066: { 10^X }
9067: function Log(X : Float) : Float; external 'dmath';
9068: { Natural log }
9069: function Log2(X : Float) : Float; external 'dmath';
9070: { Log, base 2 }
9071: function Log10(X : Float) : Float; external 'dmath';
9072: { Decimal log }
9073: function LogA(X, A : Float) : Float; external 'dmath';
9074: { Log, base A }

```

```

9075: function IntPower(X : Float; N : Integer) : Float; external 'dmath';
9076: { X^N }
9077: function Power(X, Y : Float) : Float; external 'dmath';
9078: { X^Y, X >= 0 }
9079: { -----
9080:   Trigonometric functions
9081: ----- }
9082: function Pythag(X, Y : Float) : Float; external 'dmath';
9083: { Sqrt(X^2 + Y^2) }
9084: function FixAngle(Theta : Float) : Float; external 'dmath';
9085: { Set Theta in -Pi..Pi }
9086: function Tan(X : Float) : Float; external 'dmath';
9087: { Tangent }
9088: function ArcSin(X : Float) : Float; external 'dmath';
9089: { Arc sinus }
9090: function ArcCos(X : Float) : Float; external 'dmath';
9091: { Arc cosinus }
9092: function ArcTan2(Y, X : Float) : Float; external 'dmath';
9093: { Angle (Ox, OM) with M(X,Y) }
9094: { -----
9095:   Hyperbolic functions
9096: ----- }
9097: function Sinh(X : Float) : Float; external 'dmath';
9098: { Hyperbolic sine }
9099: function Cosh(X : Float) : Float; external 'dmath';
9100: { Hyperbolic cosine }
9101: function Tanh(X : Float) : Float; external 'dmath';
9102: { Hyperbolic tangent }
9103: function ArcSinh(X : Float) : Float; external 'dmath';
9104: { Inverse hyperbolic sine }
9105: function ArcCosh(X : Float) : Float; external 'dmath';
9106: { Inverse hyperbolic cosine }
9107: function ArcTanh(X : Float) : Float; external 'dmath';
9108: { Inverse hyperbolic tangent }
9109: procedure SinhCosh(X : Float; var SinhX, CoshX : Float); external 'dmath';
9110: { Sinh & Cosh }
9111: { -----
9112:   Gamma function and related functions
9113: ----- }
9114: function Fact(N : Integer) : Float; external 'dmath';
9115: { Factorial }
9116: function SgnGamma(X : Float) : Integer; external 'dmath';
9117: { Sign of Gamma function }
9118: function Gamma(X : Float) : Float; external 'dmath';
9119: { Gamma function }
9120: function LnGamma(X : Float) : Float; external 'dmath';
9121: { Logarithm of Gamma function }
9122: function Stirling(X : Float) : Float; external 'dmath';
9123: { Stirling's formula for the Gamma function }
9124: function StirLog(X : Float) : Float; external 'dmath';
9125: { Approximate Ln(Gamma) by Stirling's formula, for X >= 13 }
9126: function DiGamma(X : Float) : Float; external 'dmath';
9127: { Digamma function }
9128: function TriGamma(X : Float) : Float; external 'dmath';
9129: { Trigamma function }
9130: function IGamma(A, X : Float) : Float; external 'dmath';
9131: { Incomplete Gamma function }
9132: function JGamma(A, X : Float) : Float; external 'dmath';
9133: { Complement of incomplete Gamma function }
9134: function InvGamma(A, P : Float) : Float; external 'dmath';
9135: { Inverse of incomplete Gamma function }
9136: function Erf(X : Float) : Float; external 'dmath';
9137: { Error function }
9138: function Erfc(X : Float) : Float; external 'dmath';
9139: { Complement of error function }
9140: { -----
9141:   Beta function and related functions
9142: ----- }
9143: function Beta(X, Y : Float) : Float; external 'dmath';
9144: { Beta function }
9145: function IBeta(A, B, X : Float) : Float; external 'dmath';
9146: { Incomplete Beta function }
9147: function InvBeta(A, B, Y : Float) : Float; external 'dmath';
9148: { Inverse of incomplete Beta function }
9149: { -----
9150:   Lambert's function
9151: ----- }
9152: function LambertW(X : Float; UBranch, Offset : Boolean) : Float; external 'dmath';
9153: { -----
9154:   Binomial distribution
9155: ----- }
9156: function Binomial(N, K : Integer) : Float; external 'dmath';
9157: { Binomial coefficient C(N,K) }
9158: function PBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9159: { Probability of binomial distribution }
9160: function FBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9161: { Cumulative probability for binomial distrib. }
9162: { -----
9163:   Poisson distribution

```

```

9164: ----- }
9165: function PPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9166: { Probability of Poisson distribution }
9167: function FPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9168: { Cumulative probability for Poisson distrib. }
9169: { -----
9170:   Exponential distribution
9171: ----- }
9172: function DExpo(A, X : Float) : Float; external 'dmath';
9173: { Density of exponential distribution with parameter A }
9174: function FExpo(A, X : Float) : Float; external 'dmath';
9175: { Cumulative probability function for exponential dist. with parameter A }
9176: { -----
9177:   Standard normal distribution
9178: ----- }
9179: function DNorm(X : Float) : Float; external 'dmath';
9180: { Density of standard normal distribution }
9181: function FNorm(X : Float) : Float; external 'dmath';
9182: { Cumulative probability for standard normal distrib. }
9183: function PNorm(X : Float) : Float; external 'dmath';
9184: { Prob(|U| > X) for standard normal distrib. }
9185: function InvNorm(P : Float) : Float; external 'dmath';
9186: { Inverse of standard normal distribution }
9187: { -----
9188:   Student's distribution
9189: ----- }
9190: function DStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9191: { Density of Student distribution with Nu d.o.f. }
9192: function FStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9193: { Cumulative probability for Student distrib. with Nu d.o.f. }
9194: function PStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9195: { Prob(|t| > X) for Student distrib. with Nu d.o.f. }
9196: function InvStudent(Nu : Integer; P : Float) : Float; external 'dmath';
9197: { Inverse of Student's t-distribution function }
9198: { -----
9199:   Khi-2 distribution
9200: ----- }
9201: function DKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9202: { Density of Khi-2 distribution with Nu d.o.f. }
9203: function FKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9204: { Cumulative prob. for Khi-2 distrib. with Nu d.o.f. }
9205: function PKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9206: { Prob(Khi2 > X) for Khi-2 distrib. with Nu d.o.f. }
9207: function InvKhi2(Nu : Integer; P : Float) : Float; external 'dmath';
9208: { Inverse of Khi-2 distribution function }
9209: { -----
9210:   Fisher-Snedecor distribution
9211: ----- }
9212: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9213: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
9214: function FSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9215: { Cumulative prob. for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9216: function PSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9217: { Prob(F > X) for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9218: function InvSnedecor(Nu1, Nu2 : Integer; P : Float) : Float; external 'dmath';
9219: { Inverse of Snedecor's F-distribution function }
9220: { -----
9221:   Beta distribution
9222: ----- }
9223: function DBeta(A, B, X : Float) : Float; external 'dmath';
9224: { Density of Beta distribution with parameters A and B }
9225: function FBeta(A, B, X : Float) : Float; external 'dmath';
9226: { Cumulative probability for Beta distrib. with param. A and B }
9227: { -----
9228:   Gamma distribution
9229: ----- }
9230: function DGamma(A, B, X : Float) : Float; external 'dmath';
9231: { Density of Gamma distribution with parameters A and B }
9232: function FGamma(A, B, X : Float) : Float; external 'dmath';
9233: { Cumulative probability for Gamma distrib. with param. A and B }
9234: { -----
9235:   Expression evaluation
9236: ----- }
9237: function InitEval : Integer; external 'dmath';
9238: { Initializes built-in functions and returns their number }
9239: function Eval(ExpressionString : String) : Float; external 'dmath';
9240: { Evaluates an expression at run-time }
9241: procedure SetVariable(VarName : Char; Value : Float); external 'dmath';
9242: { Assigns a value to a variable }
9243: procedure SetFunction(FuncName : String; Wrapper : TWrapper); external 'dmath';
9244: { Adds a function to the parser }
9245: { -----
9246:   Matrices and linear equations
9247: ----- }
9248: procedure GaussJordan(A : TMatrix;
9249:   Lb, Ub1, Ub2 : Integer;
9250:   var Det : Float); external 'dmath';
9251: { Transforms a matrix according to the Gauss-Jordan method }
9252: procedure LinEq(A : TMatrix;

```

```

9253:         B      : TVector;
9254:         Lb, Ub  : Integer;
9255:         var Det : Float); external 'dmath';
9256: { Solves a linear system according to the Gauss-Jordan method }
9257: procedure Cholesky(A, L : TMatrix; Lb, Ub : Integer); external 'dmath';
9258: { Cholesky factorization of a positive definite symmetric matrix }
9259: procedure LU_Decom(A : TMatrix; Lb, Ub : Integer); external 'dmath';
9260: { LU decomposition }
9261: procedure LU_Solve(A      : TMatrix;
9262:                   B      : TVector;
9263:                   Lb, Ub  : Integer;
9264:                   X      : TVector); external 'dmath';
9265: { Solution of linear system from LU decomposition }
9266: procedure QR_Decom(A      : TMatrix;
9267:                   Lb, Ub1, Ub2 : Integer;
9268:                   R      : TMatrix); external 'dmath';
9269: { QR decomposition }
9270: procedure QR_Solve(Q, R      : TMatrix;
9271:                   B      : TVector;
9272:                   Lb, Ub1, Ub2 : Integer;
9273:                   X      : TVector); external 'dmath';
9274: { Solution of linear system from QR decomposition }
9275: procedure SV_Decom(A      : TMatrix;
9276:                   Lb, Ub1, Ub2 : Integer;
9277:                   S      : TVector;
9278:                   V      : TMatrix); external 'dmath';
9279: { Singular value decomposition }
9280: procedure SV_SetZero(S      : TVector;
9281:                   Lb, Ub  : Integer;
9282:                   Tol      : Float); external 'dmath';
9283: { Set lowest singular values to zero }
9284: procedure SV_Solve(U      : TMatrix;
9285:                   S      : TVector;
9286:                   V      : TMatrix;
9287:                   B      : TVector;
9288:                   Lb, Ub1, Ub2 : Integer;
9289:                   X      : TVector); external 'dmath';
9290: { Solution of linear system from SVD }
9291: procedure SV_Approx(U      : TMatrix;
9292:                   S      : TVector;
9293:                   V      : TMatrix;
9294:                   Lb, Ub1, Ub2 : Integer;
9295:                   A      : TMatrix); external 'dmath';
9296: { Matrix approximation from SVD }
9297: procedure EigenVals(A      : TMatrix;
9298:                   Lb, Ub  : Integer;
9299:                   Lambda : TCompVector); external 'dmath';
9300: { Eigenvalues of a general square matrix }
9301: procedure EigenVect(A      : TMatrix;
9302:                   Lb, Ub  : Integer;
9303:                   Lambda : TCompVector;
9304:                   V      : TMatrix); external 'dmath';
9305: { Eigenvalues and eigenvectors of a general square matrix }
9306: procedure EigenSym(A      : TMatrix;
9307:                   Lb, Ub  : Integer;
9308:                   Lambda : TVector;
9309:                   V      : TMatrix); external 'dmath';
9310: { Eigenvalues and eigenvectors of a symmetric matrix (SVD method) }
9311: procedure Jacobi(A      : TMatrix;
9312:                   Lb, Ub, MaxIter : Integer;
9313:                   Tol      : Float;
9314:                   Lambda : TVector;
9315:                   V      : TMatrix); external 'dmath';
9316: { Eigenvalues and eigenvectors of a symmetric matrix (Jacobi method) }
9317: { -----
9318:   Optimization
9319:   ----- }
9320: procedure MinBrack(Func      : TFunc;
9321:                   var A, B, C, Fa, Fb, Fc : Float); external 'dmath';
9322: { Brackets a minimum of a function }
9323: procedure GoldSearch(Func      : TFunc;
9324:                   A, B      : Float;
9325:                   MaxIter    : Integer;
9326:                   Tol        : Float;
9327:                   var Xmin, Ymin : Float); external 'dmath';
9328: { Minimization of a function of one variable (golden search) }
9329: procedure LinMin(Func      : TFuncNVar;
9330:                   X, DeltaX : TVector;
9331:                   Lb, Ub    : Integer;
9332:                   var R      : Float;
9333:                   MaxIter    : Integer;
9334:                   Tol        : Float;
9335:                   var F_min  : Float); external 'dmath';
9336: { Minimization of a function of several variables along a line }
9337: procedure Newton(Func      : TFuncNVar;
9338:                   HessGrad : THessGrad;
9339:                   X      : TVector;
9340:                   Lb, Ub  : Integer;
9341:                   MaxIter : Integer;

```



```

9342:         Tol      : Float;
9343:     var F_min : Float;
9344:     G         : TVector;
9345:     H_inv     : TMatrix;
9346:     var Det   : Float); external 'dmath';
9347: { Minimization of a function of several variables (Newton's method) }
9348: procedure SaveNewton(FileName : string); external 'dmath';
9349: { Save Newton iterations in a file }
9350: procedure Marquardt(Func      : TFuncNVar;
9351:     HessGrad : THessGrad;
9352:     X         : TVector;
9353:     Lb, Ub    : Integer;
9354:     MaxIter   : Integer;
9355:     Tol       : Float;
9356:     var F_min : Float;
9357:     G         : TVector;
9358:     H_inv     : TMatrix;
9359:     var Det   : Float); external 'dmath';
9360: { Minimization of a function of several variables (Marquardt's method) }
9361: procedure SaveMarquardt(FileName : string); external 'dmath';
9362: { Save Marquardt iterations in a file }
9363: procedure BFGS(Func      : TFuncNVar;
9364:     Gradient  : TGradient;
9365:     X          : TVector;
9366:     Lb, Ub    : Integer;
9367:     MaxIter   : Integer;
9368:     Tol       : Float;
9369:     var F_min : Float;
9370:     G         : TVector;
9371:     H_inv     : TMatrix); external 'dmath';
9372: { Minimization of a function of several variables (BFGS method) }
9373: procedure SaveBFGS(FileName : string); external 'dmath';
9374: { Save BFGS iterations in a file }
9375: procedure Simplex(Func      : TFuncNVar;
9376:     X          : TVector;
9377:     Lb, Ub    : Integer;
9378:     MaxIter   : Integer;
9379:     Tol       : Float;
9380:     var F_min : Float); external 'dmath';
9381: { Minimization of a function of several variables (Simplex) }
9382: procedure SaveSimplex(FileName : string); external 'dmath';
9383: { Save Simplex iterations in a file }
9384: { -----
9385:     Nonlinear equations
9386:     ----- }
9387: procedure RootBrack(Func      : TFunc;
9388:     var X, Y, FX, FY : Float); external 'dmath';
9389: { Brackets a root of function Func between X and Y }
9390: procedure Bisection(Func      : TFunc;
9391:     var X, Y : Float;
9392:     MaxIter  : Integer;
9393:     Tol      : Float;
9394:     var F     : Float); external 'dmath';
9395: { Bisection method }
9396: procedure Secant(Func      : TFunc;
9397:     var X, Y : Float;
9398:     MaxIter  : Integer;
9399:     Tol      : Float;
9400:     var F     : Float); external 'dmath';
9401: { Secant method }
9402: procedure NewtEq(Func, Deriv : TFunc;
9403:     var X : Float;
9404:     MaxIter : Integer;
9405:     Tol     : Float;
9406:     var F   : Float); external 'dmath';
9407: { Newton-Raphson method for a single nonlinear equation }
9408: procedure NewtEqs(Equations : TEquations;
9409:     Jacobian : TJacobian;
9410:     X, F     : TVector;
9411:     Lb, Ub   : Integer;
9412:     MaxIter  : Integer;
9413:     Tol      : Float); external 'dmath';
9414: { Newton-Raphson method for a system of nonlinear equations }
9415: procedure Broyden(Equations : TEquations;
9416:     X, F : TVector;
9417:     Lb, Ub : Integer;
9418:     MaxIter : Integer;
9419:     Tol     : Float); external 'dmath';
9420: { Broyden's method for a system of nonlinear equations }
9421: { -----
9422:     Polynomials and rational fractions
9423:     ----- }
9424: function Poly(X : Float;
9425:     Coef : TVector;
9426:     Deg  : Integer) : Float; external 'dmath';
9427: { Evaluates a polynomial }
9428: function RFrac(X : Float;
9429:     Coef : TVector;
9430:     Deg1, Deg2 : Integer) : Float; external 'dmath';

```

```

9431: { Evaluates a rational fraction }
9432: function RootPol1(A, B : Float;
9433:   var X : Float) : Integer; external 'dmath';
9434: { Solves the linear equation  $A + B * X = 0$  }
9435: function RootPol2(Coef : TVector;
9436:   Z : TCompVector) : Integer; external 'dmath';
9437: { Solves a quadratic equation }
9438: function RootPol3(Coef : TVector;
9439:   Z : TCompVector) : Integer; external 'dmath';
9440: { Solves a cubic equation }
9441: function RootPol4(Coef : TVector;
9442:   Z : TCompVector) : Integer; external 'dmath';
9443: { Solves a quartic equation }
9444: function RootPol(Coef : TVector;
9445:   Deg : Integer;
9446:   Z : TCompVector) : Integer; external 'dmath';
9447: { Solves a polynomial equation }
9448: function SetRealRoots(Deg : Integer;
9449:   Z : TCompVector;
9450:   Tol : Float) : Integer; external 'dmath';
9451: { Set the imaginary part of a root to zero }
9452: procedure SortRoots(Deg : Integer;
9453:   Z : TCompVector); external 'dmath';
9454: { Sorts the roots of a polynomial }
9455: { -----
9456:   Numerical integration and differential equations
9457:   ----- }
9458: function TrapInt(X, Y : TVector; N : Integer) : Float; external 'dmath';
9459: { Integration by trapezoidal rule }
9460: function GausLeg(Func : TFunc; A, B : Float) : Float; external 'dmath';
9461: { Integral from A to B }
9462: function GausLeg0(Func : TFunc; B : Float) : Float; external 'dmath';
9463: { Integral from 0 to B }
9464: function Convolve(Func1, Func2 : TFunc; T : Float) : Float; external 'dmath';
9465: { Convolution product at time T }
9466: procedure ConvTrap(Func1, Func2 : TFunc; T, Y : TVector; N : Integer); external 'dmath';
9467: { Convolution by trapezoidal rule }
9468: procedure RKF45(F : TDiffEqs;
9469:   Negn : Integer;
9470:   Y, Yp : TVector;
9471:   var T : Float;
9472:   Tout, RelErr, AbsErr : Float;
9473:   var Flag : Integer); external 'dmath';
9474: { Integration of a system of differential equations }
9475: { -----
9476:   Fast Fourier Transform
9477:   ----- }
9478: procedure FFT(NumSamples : Integer;
9479:   InArray, OutArray : TCompVector); external 'dmath';
9480: { Fast Fourier Transform }
9481: procedure IFFT(NumSamples : Integer;
9482:   InArray, OutArray : TCompVector); external 'dmath';
9483: { Inverse Fast Fourier Transform }
9484: procedure FFT_Integer(NumSamples : Integer;
9485:   RealIn, ImagIn : TIntVector;
9486:   OutArray : TCompVector); external 'dmath';
9487: { Fast Fourier Transform for integer data }
9488: procedure FFT_Integer_Cleanup; external 'dmath';
9489: { Clear memory after a call to FFT_Integer }
9490: procedure CalcFrequency(NumSamples,
9491:   FrequencyIndex : Integer;
9492:   InArray : TCompVector;
9493:   var FFT : Complex); external 'dmath';
9494: { Direct computation of Fourier transform }
9495: { -----
9496:   Random numbers
9497:   ----- }
9498: procedure SetRNG(RNG : RNG_Type); external 'dmath';
9499: { Select generator }
9500: procedure InitGen(Seed : RNG_IntType); external 'dmath';
9501: { Initialize generator }
9502: function IRanGen : RNG_IntType; external 'dmath';
9503: { 32-bit random integer in  $[-2^{31} .. 2^{31} - 1]$  }
9504: function IRanGen31 : RNG_IntType; external 'dmath';
9505: { 31-bit random integer in  $[0 .. 2^{31} - 1]$  }
9506: function RanGen1 : Float; external 'dmath';
9507: { 32-bit random real in  $[0,1]$  }
9508: function RanGen2 : Float; external 'dmath';
9509: { 32-bit random real in  $(0,1)$  }
9510: function RanGen3 : Float; external 'dmath';
9511: { 32-bit random real in  $(0,1)$  }
9512: function RanGen53 : Float; external 'dmath';
9513: { 53-bit random real in  $[0,1]$  }
9514: procedure InitMWC(Seed : RNG_IntType); external 'dmath';
9515: { Initializes the 'Multiply with carry' random number generator }
9516: function IRanMWC : RNG_IntType; external 'dmath';
9517: { Returns a 32 bit random number in  $[-2^{31} .. 2^{31}-1]$  }
9518: procedure InitMT(Seed : RNG_IntType); external 'dmath';
9519: { Initializes Mersenne Twister generator with a seed }

```

```

9520: procedure InitMTbyArray(InitKey   : array of RNG_LongType;
9521:                        KeyLength : Word); external 'dmath';
9522: { Initialize MT generator with an array InitKey[0..(KeyLength - 1)] }
9523: function IRanMT : RNG_IntType; external 'dmath';
9524: { Random integer from MT generator }
9525: procedure InitUVAGbyString(KeyPhrase : string); external 'dmath';
9526: { Initializes the UVAG generator with a string }
9527: procedure InitUVAG(Seed : RNG_IntType); external 'dmath';
9528: { Initializes the UVAG generator with an integer }
9529: function IRanUVAG : RNG_IntType; external 'dmath';
9530: { Random integer from UVAG generator }
9531: function RanGaussStd : Float; external 'dmath';
9532: { Random number from standard normal distribution }
9533: function RanGauss(Mu, Sigma : Float) : Float; external 'dmath';
9534: { Random number from normal distrib. with mean Mu and S. D. Sigma }
9535: procedure RanMult(M      : TVector; L      : TMatrix;
9536:                  Lb, Ub : Integer;
9537:                  X      : TVector); external 'dmath';
9538: { Random vector from multinormal distribution (correlated) }
9539: procedure RanMultIndep(M, S : TVector;
9540:                       Lb, Ub : Integer;
9541:                       X      : TVector); external 'dmath';
9542: { Random vector from multinormal distribution (uncorrelated) }
9543: procedure InitMHParams(NCycles, MaxSim, SavedSim : Integer); external 'dmath';
9544: { Initializes Metropolis-Hastings parameters }
9545: procedure GetMHParams(var NCycles, MaxSim, SavedSim: Integer); external 'dmath';
9546: { Returns Metropolis-Hastings parameters }
9547: procedure Hastings(Func      : TFuncNVar;
9548:                   T         : Float;
9549:                   X         : TVector;
9550:                   V         : TMatrix;
9551:                   Lb, Ub    : Integer;
9552:                   Xmat     : TMatrix;
9553:                   X_min    : TVector;
9554:                   var F_min : Float); external 'dmath';
9555: { Simulation of a probability density function by Metropolis-Hastings }
9556: procedure InitSAParams(NT, NS, NCycles : Integer; RT : Float); external 'dmath';
9557: { Initializes Simulated Annealing parameters }
9558: procedure SA_CreateLogFile(FileName : String); external 'dmath';
9559: { Initializes log file }
9560: procedure SimAnn(Func      : TFuncNVar;
9561:                 X, Xmin, Xmax : TVector;
9562:                 Lb, Ub      : Integer;
9563:                 var F_min    : Float); external 'dmath';
9564: { Minimization of a function of several var. by simulated annealing }
9565: procedure InitGAParams(NP, NG : Integer; SR, MR, HR : Float); external 'dmath';
9566: { Initializes Genetic Algorithm parameters }
9567: procedure GA_CreateLogFile(FileName : String); external 'dmath';
9568: { Initializes log file }
9569: procedure GenAlg(Func      : TFuncNVar;
9570:                 X, Xmin, Xmax : TVector;
9571:                 Lb, Ub      : Integer;
9572:                 var F_min    : Float); external 'dmath';
9573: { Minimization of a function of several var. by genetic algorithm }
9574: { -----
9575:   Statistics
9576:   ----- }
9577: function Mean(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9578: { Mean of sample X }
9579: function Min(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9580: { Minimum of sample X }
9581: function Max(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9582: { Maximum of sample X }
9583: function Median(X : TVector; Lb, Ub : Integer; Sorted : Boolean) : Float; external 'dmath';
9584: { Median of sample X }
9585: function StDev(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9586: { Standard deviation estimated from sample X }
9587: function StDevP(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9588: { Standard deviation of population }
9589: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9590: { Correlation coefficient }
9591: function Skewness(X : TVector; Lb, Ub : Integer; M, Sigma: Float) : Float; external 'dmath';
9592: { Skewness of sample X }
9593: function Kurtosis(X : TVector; Lb, Ub : Integer; M, Sigma: Float) : Float; external 'dmath';
9594: { Kurtosis of sample X }
9595: procedure QSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9596: { Quick sort (ascending order) }
9597: procedure DQSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9598: { Quick sort (descending order) }
9599: procedure Interval(X1, X2      : Float;
9600:                  MinDiv, MaxDiv : Integer;
9601:                  var Min, Max, Step : Float); external 'dmath';
9602: { Determines an interval for a set of values }
9603: procedure AutoScale(X : TVector; Lb, Ub : Integer; Scale : TScale;
9604:                   var XMin, XMax, XStep : Float); external 'dmath';
9605: { Finds an appropriate scale for plotting the data in X[Lb..Ub] }
9606: procedure StudIndep(N1, N2      : Integer;
9607:                   M1, M2, S1, S2 : Float;
9608:                   var T          : Float);

```

```

9609:         var DoF          : Integer); external 'dmath';
9610: { Student t-test for independent samples }
9611: procedure StudPaired(X, Y      : TVector;
9612:                    Lb, Ub     : Integer;
9613:                    var T       : Float;
9614:                    var DoF     : Integer); external 'dmath';
9615: { Student t-test for paired samples }
9616: procedure AnOVA1(Ns          : Integer;
9617:                N            : TIntVector;
9618:                M, S         : TVector;
9619:                var V_f, V_r, F : Float;
9620:                var DoF_f, DoF_r : Integer); external 'dmath';
9621: { One-way analysis of variance }
9622: procedure AnOVA2(NA, NB, Nobs : Integer;
9623:                M, S         : TMatrix;
9624:                V, F         : TVector;
9625:                DoF          : TIntVector); external 'dmath';
9626: { Two-way analysis of variance }
9627: procedure Snedecor(N1, N2      : Integer;
9628:                  S1, S2       : Float;
9629:                  var F         : Float;
9630:                  var DoF1, DoF2 : Integer); external 'dmath';
9631: { Snedecor's F-test (comparison of two variances) }
9632: procedure Bartlett(Ns          : Integer;
9633:                  N            : TIntVector;
9634:                  S            : TVector;
9635:                  var Khi2      : Float;
9636:                  var DoF       : Integer); external 'dmath';
9637: { Bartlett's test (comparison of several variances) }
9638: procedure Mann_Whitney(N1, N2   : Integer;
9639:                      X1, X2     : TVector;
9640:                      var U, Eps  : Float); external 'dmath';
9641: { Mann-Whitney test }
9642: procedure Wilcoxon(X, Y        : TVector;
9643:                  Lb, Ub       : Integer;
9644:                  var Ndiff     : Integer;
9645:                  var T, Eps    : Float); external 'dmath';
9646: { Wilcoxon test }
9647: procedure Kruskal_Wallis(Ns      : Integer;
9648:                         N        : TIntVector;
9649:                         X        : TMatrix;
9650:                         var H     : Float;
9651:                         var DoF   : Integer); external 'dmath';
9652: { Kruskal-Wallis test }
9653: procedure Khi2_Conform(N_cls     : Integer;
9654:                      N_estim     : Integer;
9655:                      Obs         : TIntVector;
9656:                      Calc        : TVector;
9657:                      var Khi2     : Float;
9658:                      var DoF      : Integer); external 'dmath';
9659: { Khi-2 test for conformity }
9660: procedure Khi2_Indep(N_lin       : Integer;
9661:                    N_col        : Integer;
9662:                    Obs          : TIntMatrix;
9663:                    var Khi2      : Float;
9664:                    var DoF       : Integer); external 'dmath';
9665: { Khi-2 test for independence }
9666: procedure Woolf_Conform(N_cls     : Integer;
9667:                      N_estim     : Integer;
9668:                      Obs         : TIntVector;
9669:                      Calc        : TVector;
9670:                      var G       : Float;
9671:                      var DoF     : Integer); external 'dmath';
9672: { Woolf's test for conformity }
9673: procedure Woolf_Indep(N_lin       : Integer;
9674:                    N_col        : Integer;
9675:                    Obs          : TIntMatrix;
9676:                    var G       : Float;
9677:                    var DoF     : Integer); external 'dmath';
9678: { Woolf's test for independence }
9679: procedure DimStatClassVector(var C : TStatClassVector;
9680:                             Ub    : Integer); external 'dmath';
9681: { Allocates an array of statistical classes: C[0..Ub] }
9682: procedure Distrib(X          : TVector;
9683:                 Lb, Ub      : Integer;
9684:                 A, B, H     : Float;
9685:                 C           : TStatClassVector); external 'dmath';
9686: { Distributes an array X[Lb..Ub] into statistical classes }
9687: { -----
9688:   Linear / polynomial regression
9689:   ----- }
9690: procedure LinFit(X, Y      : TVector;
9691:                Lb, Ub     : Integer;
9692:                B          : TVector;
9693:                V          : TMatrix); external 'dmath';
9694: { Linear regression : Y = B(0) + B(1) * X }
9695: procedure WLinFit(X, Y, S : TVector;
9696:                 Lb, Ub   : Integer;
9697:                 B        : TVector;

```



```

9698:         V          : TMatrix); external 'dmath';
9699: { Weighted linear regression :  $Y = B(0) + B(1) * X$  }
9700: procedure SVDLinFit(X, Y : TVector;
9701:                   Lb, Ub : Integer;
9702:                   SVDTol : Float;
9703:                   B      : TVector;
9704:                   V      : TMatrix); external 'dmath';
9705: { Unweighted linear regression by singular value decomposition }
9706: procedure WSVDLinFit(X, Y, S : TVector;
9707:                    Lb, Ub : Integer;
9708:                    SVDTol : Float;
9709:                    B      : TVector;
9710:                    V      : TMatrix); external 'dmath';
9711: { Weighted linear regression by singular value decomposition }
9712: procedure MulFit(X          : TMatrix;
9713:                Y          : TVector;
9714:                Lb, Ub, Nvar : Integer;
9715:                ConsTerm    : Boolean;
9716:                B           : TVector;
9717:                V           : TMatrix); external 'dmath';
9718: { Multiple linear regression by Gauss-Jordan method }
9719: procedure WMulFit(X          : TMatrix;
9720:                 Y, S        : TVector;
9721:                 Lb, Ub, Nvar : Integer;
9722:                 ConsTerm    : Boolean;
9723:                 B           : TVector;
9724:                 V           : TMatrix); external 'dmath';
9725: { Weighted multiple linear regression by Gauss-Jordan method }
9726: procedure SVDFit(X          : TMatrix;
9727:                Y          : TVector;
9728:                Lb, Ub, Nvar : Integer;
9729:                ConsTerm    : Boolean;
9730:                SVDTol      : Float;
9731:                B           : TVector;
9732:                V           : TMatrix); external 'dmath';
9733: { Multiple linear regression by singular value decomposition }
9734: procedure WSVDFit(X          : TMatrix;
9735:                 Y, S        : TVector;
9736:                 Lb, Ub, Nvar : Integer;
9737:                 ConsTerm    : Boolean;
9738:                 SVDTol      : Float;
9739:                 B           : TVector;
9740:                 V           : TMatrix); external 'dmath';
9741: { Weighted multiple linear regression by singular value decomposition }
9742: procedure PolFit(X, Y      : TVector;
9743:                Lb, Ub, Deg : Integer;
9744:                B          : TVector;
9745:                V          : TMatrix); external 'dmath';
9746: { Polynomial regression by Gauss-Jordan method }
9747: procedure WPolFit(X, Y, S : TVector;
9748:                 Lb, Ub, Deg : Integer;
9749:                 B          : TVector;
9750:                 V          : TMatrix); external 'dmath';
9751: { Weighted polynomial regression by Gauss-Jordan method }
9752: procedure SVDPolFit(X, Y      : TVector;
9753:                  Lb, Ub, Deg : Integer;
9754:                  SVDTol      : Float;
9755:                  B          : TVector;
9756:                  V          : TMatrix); external 'dmath';
9757: { Unweighted polynomial regression by singular value decomposition }
9758: procedure WSVDPolFit(X, Y, S : TVector;
9759:                   Lb, Ub, Deg : Integer;
9760:                   SVDTol      : Float;
9761:                   B          : TVector;
9762:                   V          : TMatrix); external 'dmath';
9763: { Weighted polynomial regression by singular value decomposition }
9764: procedure RegTest(Y, Ycalc : TVector;
9765:                 LbY, UbY : Integer;
9766:                 V        : TMatrix;
9767:                 LbV, UbV : Integer;
9768:                 var Test : TRegTest); external 'dmath';
9769: { Test of unweighted regression }
9770: procedure WRegTest(Y, Ycalc, S : TVector;
9771:                  LbY, UbY : Integer;
9772:                  V        : TMatrix;
9773:                  LbV, UbV : Integer;
9774:                  var Test : TRegTest); external 'dmath';
9775: { Test of weighted regression }
9776: { -----
9777:   Nonlinear regression
9778:   ----- }
9779: procedure SetOptAlgo(Algo : TOptAlgo); external 'dmath';
9780: { Sets the optimization algorithm for nonlinear regression }
9781: function GetOptAlgo : TOptAlgo; external 'dmath';
9782: { Returns the optimization algorithm }
9783: procedure SetMaxParam(N : Byte); external 'dmath';
9784: { Sets the maximum number of regression parameters for nonlinear regression }
9785: function GetMaxParam : Byte; external 'dmath';
9786: { Returns the maximum number of regression parameters for nonlinear regression }

```

```

9787: procedure SetParamBounds(I : Byte; ParamMin, ParamMax : Float); external 'dmath';
9788: { Sets the bounds on the I-th regression parameter }
9789: procedure GetParamBounds(I : Byte; var ParamMin, ParamMax:Float); external 'dmath';
9790: { Returns the bounds on the I-th regression parameter }
9791: procedure NLFit(RegFunc : TRegFunc;
9792:               DerivProc : TDerivProc;
9793:               X, Y       : TVector;
9794:               Lb, Ub     : Integer;
9795:               MaxIter    : Integer;
9796:               Tol        : Float;
9797:               B          : TVector;
9798:               FirstPar,
9799:               LastPar    : Integer;
9800:               V          : TMatrix); external 'dmath';
9801: { Unweighted nonlinear regression }
9802: procedure WNLFit(RegFunc : TRegFunc;
9803:                DerivProc : TDerivProc;
9804:                X, Y, S    : TVector;
9805:                Lb, Ub    : Integer;
9806:                MaxIter   : Integer;
9807:                Tol       : Float;
9808:                B         : TVector;
9809:                FirstPar,
9810:                LastPar   : Integer;
9811:                V         : TMatrix); external 'dmath';
9812: { Weighted nonlinear regression }
9813: procedure SetMCFile(FileName : String); external 'dmath';
9814: { Set file for saving MCMC simulations }
9815: procedure SimFit(RegFunc : TRegFunc;
9816:                 X, Y      : TVector;
9817:                 Lb, Ub    : Integer;
9818:                 B         : TVector;
9819:                 FirstPar,
9820:                 LastPar   : Integer;
9821:                 V         : TMatrix); external 'dmath';
9822: { Simulation of unweighted nonlinear regression by MCMC }
9823: procedure WSimFit(RegFunc : TRegFunc;
9824:                  X, Y, S   : TVector;
9825:                  Lb, Ub   : Integer;
9826:                  B        : TVector;
9827:                  FirstPar,
9828:                  LastPar  : Integer;
9829:                  V        : TMatrix); external 'dmath';
9830: { Simulation of weighted nonlinear regression by MCMC }
9831: { -----
9832:   Nonlinear regression models
9833:   ----- }
9834: procedure FracFit(X, Y      : TVector;
9835:                  Lb, Ub    : Integer;
9836:                  Deg1, Deg2 : Integer;
9837:                  ConstTerm  : Boolean;
9838:                  MaxIter    : Integer;
9839:                  Tol        : Float;
9840:                  B          : TVector;
9841:                  V          : TMatrix); external 'dmath';
9842: { Unweighted fit of rational fraction }
9843: procedure WFracFit(X, Y, S : TVector;
9844:                   Lb, Ub   : Integer;
9845:                   Deg1, Deg2 : Integer;
9846:                   ConstTerm : Boolean;
9847:                   MaxIter   : Integer;
9848:                   Tol       : Float;
9849:                   B         : TVector;
9850:                   V         : TMatrix); external 'dmath';
9851: { Weighted fit of rational fraction }
9852:
9853: function FracFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9854: { Returns the value of the rational fraction at point X }
9855: procedure ExpFit(X, Y      : TVector;
9856:                  Lb, Ub, Nexp : Integer;
9857:                  ConstTerm    : Boolean;
9858:                  MaxIter      : Integer;
9859:                  Tol          : Float;
9860:                  B            : TVector;
9861:                  V            : TMatrix); external 'dmath';
9862: { Unweighted fit of sum of exponentials }
9863: procedure WExpFit(X, Y, S : TVector;
9864:                  Lb, Ub, Nexp : Integer;
9865:                  ConstTerm    : Boolean;
9866:                  MaxIter      : Integer;
9867:                  Tol          : Float;
9868:                  B            : TVector;
9869:                  V            : TMatrix); external 'dmath';
9870: { Weighted fit of sum of exponentials }
9871: function ExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9872: { Returns the value of the regression function at point X }
9873: procedure IncExpFit(X, Y      : TVector;
9874:                    Lb, Ub    : Integer;
9875:                    ConstTerm : Boolean;

```

```

9876:           MaxIter : Integer;
9877:           Tol      : Float;
9878:           B        : TVector;
9879:           V        : TMatrix); external 'dmath';
9880: { Unweighted fit of model of increasing exponential }
9881: procedure WIncExpFit(X, Y, S : TVector;
9882:           Lb, Ub : Integer;
9883:           ConstTerm : Boolean;
9884:           MaxIter : Integer;
9885:           Tol      : Float;
9886:           B        : TVector;
9887:           V        : TMatrix); external 'dmath';
9888: { Weighted fit of increasing exponential }
9889: function IncExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9890: { Returns the value of the regression function at point X }
9891: procedure ExpLinFit(X, Y : TVector;
9892:           Lb, Ub : Integer;
9893:           MaxIter : Integer;
9894:           Tol      : Float;
9895:           B        : TVector;
9896:           V        : TMatrix); external 'dmath';
9897: { Unweighted fit of the "exponential + linear" model }
9898: procedure WExpLinFit(X, Y, S : TVector;
9899:           Lb, Ub : Integer;
9900:           MaxIter : Integer;
9901:           Tol      : Float;
9902:           B        : TVector;
9903:           V        : TMatrix); external 'dmath';
9904: { Weighted fit of the "exponential + linear" model }
9905:
9906: function ExpLinFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9907: { Returns the value of the regression function at point X }
9908: procedure MichFit(X, Y : TVector;
9909:           Lb, Ub : Integer;
9910:           MaxIter : Integer;
9911:           Tol      : Float;
9912:           B        : TVector;
9913:           V        : TMatrix); external 'dmath';
9914: { Unweighted fit of Michaelis equation }
9915: procedure WMichFit(X, Y, S : TVector;
9916:           Lb, Ub : Integer;
9917:           MaxIter : Integer;
9918:           Tol      : Float;
9919:           B        : TVector;
9920:           V        : TMatrix); external 'dmath';
9921: { Weighted fit of Michaelis equation }
9922: function MichFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9923: { Returns the value of the Michaelis equation at point X }
9924: procedure MintFit(X, Y : TVector;
9925:           Lb, Ub : Integer;
9926:           MintVar : TMintVar;
9927:           Fit_S0 : Boolean;
9928:           MaxIter : Integer;
9929:           Tol      : Float;
9930:           B        : TVector;
9931:           V        : TMatrix); external 'dmath';
9932: { Unweighted fit of the integrated Michaelis equation }
9933: procedure WMintFit(X, Y, S : TVector;
9934:           Lb, Ub : Integer;
9935:           MintVar : TMintVar;
9936:           Fit_S0 : Boolean;
9937:           MaxIter : Integer;
9938:           Tol      : Float;
9939:           B        : TVector;
9940:           V        : TMatrix); external 'dmath';
9941: { Weighted fit of the integrated Michaelis equation }
9942: function MintFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9943: { Returns the value of the integrated Michaelis equation at point X }
9944: procedure HillFit(X, Y : TVector;
9945:           Lb, Ub : Integer;
9946:           MaxIter : Integer;
9947:           Tol      : Float;
9948:           B        : TVector;
9949:           V        : TMatrix); external 'dmath';
9950: { Unweighted fit of Hill equation }
9951: procedure WHillFit(X, Y, S : TVector;
9952:           Lb, Ub : Integer;
9953:           MaxIter : Integer;
9954:           Tol      : Float;
9955:           B        : TVector;
9956:           V        : TMatrix); external 'dmath';
9957: { Weighted fit of Hill equation }
9958: function HillFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9959: { Returns the value of the Hill equation at point X }
9960: procedure LogiFit(X, Y : TVector;
9961:           Lb, Ub : Integer;
9962:           ConstTerm : Boolean;
9963:           General : Boolean;
9964:           MaxIter : Integer;

```

```

9965:         Tol      : Float;
9966:         B         : TVector;
9967:         V         : TMatrix); external 'dmath';
9968: { Unweighted fit of logistic function }
9969: procedure WLogiFit(X, Y, S : TVector;
9970:         Lb, Ub      : Integer;
9971:         ConstTerm    : Boolean;
9972:         General       : Boolean;
9973:         MaxIter       : Integer;
9974:         Tol           : Float;
9975:         B             : TVector;
9976:         V             : TMatrix); external 'dmath';
9977: { Weighted fit of logistic function }
9978: function LogiFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9979: { Returns the value of the logistic function at point X }
9980: procedure PKFit(X, Y : TVector;
9981:         Lb, Ub      : Integer;
9982:         MaxIter      : Integer;
9983:         Tol          : Float;
9984:         B            : TVector;
9985:         V            : TMatrix); external 'dmath';
9986: { Unweighted fit of the acid-base titration curve }
9987: procedure WPKFit(X, Y, S : TVector;
9988:         Lb, Ub      : Integer;
9989:         MaxIter      : Integer;
9990:         Tol          : Float;
9991:         B            : TVector;
9992:         V            : TMatrix); external 'dmath';
9993: { Weighted fit of the acid-base titration curve }
9994: function PKFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9995: { Returns the value of the acid-base titration function at point X }
9996: procedure PowFit(X, Y : TVector;
9997:         Lb, Ub      : Integer;
9998:         MaxIter      : Integer;
9999:         Tol          : Float;
10000:         B            : TVector;
10001:         V            : TMatrix); external 'dmath';
10002: { Unweighted fit of power function }
10003: procedure WPowFit(X, Y, S : TVector;
10004:         Lb, Ub      : Integer;
10005:         MaxIter      : Integer;
10006:         Tol          : Float;
10007:         B            : TVector;
10008:         V            : TMatrix); external 'dmath';
10009: { Weighted fit of power function }
10010:
10011: function PowFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10012: { Returns the value of the power function at point X }
10013: procedure GammaFit(X, Y : TVector;
10014:         Lb, Ub      : Integer;
10015:         MaxIter      : Integer;
10016:         Tol          : Float;
10017:         B            : TVector;
10018:         V            : TMatrix); external 'dmath';
10019: { Unweighted fit of gamma distribution function }
10020: procedure WGammaFit(X, Y, S : TVector;
10021:         Lb, Ub      : Integer;
10022:         MaxIter      : Integer;
10023:         Tol          : Float;
10024:         B            : TVector;
10025:         V            : TMatrix); external 'dmath';
10026: { Weighted fit of gamma distribution function }
10027: function GammaFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10028: { Returns the value of the gamma distribution function at point X }
10029: { -----
10030:   Principal component analysis
10031:   ----- }
10032: procedure VecMean(X : TMatrix;
10033:         Lb, Ub, Nvar : Integer;
10034:         M             : TVector); external 'dmath';
10035: { Computes the mean vector M from matrix X }
10036: procedure VecSD(X : TMatrix;
10037:         Lb, Ub, Nvar : Integer;
10038:         M, S         : TVector); external 'dmath';
10039: { Computes the vector of standard deviations S from matrix X }
10040: procedure MatVarCov(X : TMatrix;
10041:         Lb, Ub, Nvar : Integer;
10042:         M             : TVector;
10043:         V             : TMatrix); external 'dmath';
10044: { Computes the variance-covariance matrix V from matrix X }
10045: procedure MatCorrel(V : TMatrix;
10046:         Nvar : Integer;
10047:         R     : TMatrix); external 'dmath';
10048: { Computes the correlation matrix R from the var-cov matrix V }
10049: procedure PCA(R : TMatrix;
10050:         Nvar : Integer;
10051:         Lambda : TVector;
10052:         C, Rc : TMatrix); external 'dmath';
10053: { Performs a principal component analysis of the correlation matrix R }

```



```

10054: procedure ScaleVar(X           : TMatrix;
10055:                    Lb, Ub, Nvar : Integer;
10056:                    M, S         : TVector;
10057:                    Z           : TMatrix); external 'dmath';
10058: { Scales a set of variables by subtracting means and dividing by SD's }
10059: procedure PrinFac(Z           : TMatrix;
10060:                  Lb, Ub, Nvar : Integer;
10061:                  C, F         : TMatrix); external 'dmath';
10062: { Computes principal factors }
10063: { ----- }
10064:   Strings
10065: { ----- }
10066: function LTrim(S : String) : String; external 'dmath';
10067: { Removes leading blanks }
10068: function RTrim(S : String) : String; external 'dmath';
10069: { Removes trailing blanks }
10070: function Trim(S : String) : String; external 'dmath';
10071: { Removes leading and trailing blanks }
10072: function StrChar(N : Byte; C : Char) : String; external 'dmath';
10073: { Returns a string made of character C repeated N times }
10074: function RFill(S : String; L : Byte) : String; external 'dmath';
10075: { Completes string S with trailing blanks for a total length L }
10076: function LFill(S : String; L : Byte) : String; external 'dmath';
10077: { Completes string S with leading blanks for a total length L }
10078: function CFill(S : String; L : Byte) : String; external 'dmath';
10079: { Centers string S on a total length L }
10080: function Replace(S : String; C1, C2 : Char) : String; external 'dmath';
10081: { Replaces in string S all the occurrences of C1 by C2 }
10082: function Extract(S : String; var Index : Byte; Delim : Char) : String; external 'dmath';
10083: { Extracts a field from a string }
10084: procedure Parse(S : String; Delim:Char; Field:TStrVector; var N:Byte); external 'dmath';
10085: { Parses a string into its constitutive fields }
10086: procedure SetFormat(NumLength,MaxDec:Integer;FloatPoint,NSZero:Bool); external 'dmath';
10087: { Sets the numeric format }
10088: function FloatStr(X : Float) : String; external 'dmath';
10089: { Converts a real to a string according to the numeric format }
10090: function IntStr(N : LongInt) : String; external 'dmath';
10091: { Converts an integer to a string }
10092: function CompStr(Z : Complex) : String; external 'dmath';
10093: { Converts a complex number to a string }
10094: {$IFDEF DELPHI}
10095: function StrDec(S : String) : String; external 'dmath';
10096: { Set decimal separator to the symbol defined in SysUtils }
10097: function IsNumeric(var S : String; var X : Float) : Boolean; external 'dmath';
10098: { Test if a string represents a number and returns it in X }
10099: function ReadNumFromEdit(Edit : TEdit) : Float; external 'dmath';
10100: { Reads a floating point number from an Edit control }
10101: procedure WriteNumToFile(var F : Text; X : Float); external 'dmath';
10102: { Writes a floating point number in a text file }
10103: {$ENDIF}
10104: { ----- }
10105:   BGI / Delphi graphics
10106: { ----- }
10107: function InitGraphics
10108: {$IFDEF DELPHI}
10109: (Width, Height : Integer) : Boolean;
10110: {$ELSE}
10111: (Pilot, Mode : Integer; BGIPath : String) : Boolean; {$ENDIF} external 'dmath';
10112: { Enters graphic mode }
10113: procedure SetWindow({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10114:                   X1, X2, Y1,Y2 : Integer; GraphBorder:Boolean); external 'dmath';
10115: { Sets the graphic window }
10116: procedure SetOxScale(Scale           : TScale;
10117:                    OxMin, OxMax, OxStep : Float); external 'dmath';
10118: { Sets the scale on the Ox axis }
10119: procedure SetOyScale(Scale           : TScale;
10120:                    OyMin, OyMax, OyStep : Float); external 'dmath';
10121: { Sets the scale on the Oy axis }
10122: procedure GetOxScale(var Scale           : TScale;
10123:                    var OxMin, OxMax, OxStep : Float); external 'dmath';
10124: { Returns the scale on the Ox axis }
10125: procedure GetOyScale(var Scale           : TScale;
10126:                    var OyMin, OyMax, OyStep : Float); external 'dmath';
10127: { Returns the scale on the Oy axis }
10128: procedure SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10129: procedure SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
10130: procedure SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10131: function GetGraphTitle : String; external 'dmath'; { Returns the title for the graph }
10132: function GetOxTitle : String; external 'dmath'; { Returns the title for the Ox axis }
10133: function GetOyTitle : String; external 'dmath'; { Returns the title for the Oy axis }
10134: {$IFDEF DELPHI}
10135: procedure SetTitleFont(FontIndex, Width, Height : Integer); external 'dmath';
10136: { Sets the font for the main graph title }
10137: procedure SetOxFont(FontIndex, Width, Height : Integer); external 'dmath';
10138: { Sets the font for the Ox axis (title and labels) }
10139: procedure SetOyFont(FontIndex, Width, Height : Integer); external 'dmath';
10140: { Sets the font for the Oy axis (title and labels) }
10141: procedure SetLgdFont(FontIndex, Width, Height : Integer); external 'dmath';
10142: { Sets the font for the legends }

```

```

10143: procedure SetClipping(Clip : Boolean); external 'dmath';
10144: { Determines whether drawings are clipped at the current viewport
10145:   boundaries, according to the value of the Boolean parameter Clip }
10146: { $ENDIF }
10147: procedure PlotOxAxis({ $IFDEF DELPHI }Canvas : TCanvas){ $ENDIF } external 'dmath';
10148: { Plots the horizontal axis }
10149: procedure PlotOyAxis({ $IFDEF DELPHI }Canvas : TCanvas){ $ENDIF } external 'dmath';
10150: { Plots the vertical axis }
10151: procedure PlotGrid({ $IFDEF DELPHI }Canvas: TCanvas; { $ENDIF } Grid: TGrid); external 'dmath';
10152: { Plots a grid on the graph }
10153: procedure WriteGraphTitle({ $IFDEF DELPHI }Canvas : TCanvas){ $ENDIF } external 'dmath';
10154: { Writes the title of the graph }
10155: procedure SetMaxCurv(NCurv : Byte); external 'dmath';
10156: { Sets the maximum number of curves and re-initializes their parameters }
10157: procedure SetPointParam
10158:   { $IFDEF DELPHI }
10159:   (CurvIndex, Symbol, Size : Integer; Color : TColor);
10160:   { $ELSE }
10161:   (CurvIndex, Symbol, Size, Color : Integer); { $ENDIF } external 'dmath';
10162: { Sets the point parameters for curve # CurvIndex }
10163: procedure SetLineParam
10164:   { $IFDEF DELPHI }
10165:   (CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor);
10166:   { $ELSE }
10167:   (CurvIndex, Style, Width, Color : Integer); { $ENDIF } external 'dmath';
10168: { Sets the line parameters for curve # CurvIndex }
10169: procedure SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10170: { Sets the legend for curve # CurvIndex }
10171: procedure SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10172: { Sets the step for curve # CurvIndex }
10173: function GetMaxCurv : Byte; external 'dmath'; { Returns the maximum number of curves }
10174: procedure GetPointParam
10175:   { $IFDEF DELPHI }
10176:   (CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor);
10177:   { $ELSE }
10178:   (CurvIndex : Integer; var Symbol, Size, Color : Integer); { $ENDIF } external 'dmath';
10179: { Returns the point parameters for curve # CurvIndex }
10180: procedure GetLineParam
10181:   { $IFDEF DELPHI }
10182:   (CurvIndex : Integer; var Style : TPenStyle; var Width : Integer; var Color : TColor);
10183:   { $ELSE }
10184:   (CurvIndex : Integer; var Style, Width, Color : Integer); { $ENDIF } external 'dmath';
10185: { Returns the line parameters for curve # CurvIndex }
10186: function GetCurvLegend(CurvIndex : Integer) : String; external 'dmath';
10187: { Returns the legend for curve # CurvIndex }
10188: function GetCurvStep(CurvIndex : Integer) : Integer; external 'dmath';
10189: { Returns the step for curve # CurvIndex }
10190: { $IFDEF DELPHI }
10191: procedure PlotPoint(Canvas : TCanvas;
10192:   X, Y : Float; CurvIndex : Integer); external 'dmath';
10193: { $ELSE }
10194: procedure PlotPoint(Xp, Yp, CurvIndex : Integer); external 'dmath';
10195: { $ENDIF }
10196: { Plots a point on the screen }
10197: procedure PlotCurve({ $IFDEF DELPHI }Canvas : TCanvas; { $ENDIF }
10198:   X, Y : TVector;
10199:   Lb, Ub, CurvIndex : Integer); external 'dmath';
10200: { Plots a curve }
10201: procedure PlotCurveWithErrorBars({ $IFDEF DELPHI }Canvas : TCanvas; { $ENDIF }
10202:   X, Y, S : TVector;
10203:   Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10204: { Plots a curve with error bars }
10205: procedure PlotFunc({ $IFDEF DELPHI }Canvas : TCanvas; { $ENDIF }
10206:   Func : TFunc;
10207:   Xmin, Xmax : Float;
10208:   { $IFDEF DELPHI }Npt : Integer; { $ENDIF }
10209:   CurvIndex : Integer); external 'dmath';
10210: { Plots a function }
10211: procedure WriteLegend({ $IFDEF DELPHI }Canvas : TCanvas; { $ENDIF }
10212:   NCurv : Integer;
10213:   ShowPoints, ShowLines : Boolean); external 'dmath';
10214: { Writes the legends for the plotted curves }
10215: procedure ConRec({ $IFDEF DELPHI }Canvas : TCanvas; { $ENDIF }
10216:   Nx, Ny, Nc : Integer;
10217:   X, Y, Z : TVector;
10218:   F : TMatrix); external 'dmath';
10219: { Contour plot }
10220: function Xpixel(X : Float):Integer; external 'dmath'; { Converts user abscissa X to screen coordinate }
10221: function Ypixel(Y : Float):Integer; external 'dmath'; { Converts user ordinate Y to screen coordinate }
10222: function Xuser(X : Integer):Float; external 'dmath'; { Converts screen coordinate X to user abscissa }
10223: function Yuser(Y : Integer):Float; external 'dmath'; { Converts screen coordinate Y to user ordinate }
10224: { $IFDEF DELPHI }
10225: procedure LeaveGraphics; external 'dmath';
10226: { Quits graphic mode }
10227: { $ENDIF }
10228: { -----
10229:   LaTeX graphics
10230:   ----- }
10231: function TeX_InitGraphics(FileName : String; PgWidth, PgHeight : Integer;

```

```

10232:                                     Header                               : Boolean) : Boolean; external 'dmath';
10233: { Initializes the LaTeX file }
10234: procedure TeX_SetWindow(X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean); external 'dmath';
10235: { Sets the graphic window }
10236: procedure TeX_LeaveGraphics(Footer : Boolean); external 'dmath'; { Close the LaTeX file }
10237: procedure TeX_SetOxScale(Scale : TScale; OxMin, OxMax, OxStep : Float); external 'dmath';
10238: { Sets the scale on the Ox axis }
10239: procedure TeX_SetOyScale(Scale : TScale; OyMin, OyMax, OyStep : Float); external 'dmath';
10240: { Sets the scale on the Oy axis }
10241: procedure TeX_SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10242: procedure TeX_SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
10243: procedure TeX_SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10244: procedure TeX_PlotOxAxis; external 'dmath'; { Plots the horizontal axis }
10245: procedure TeX_PlotOyAxis; external 'dmath'; { Plots the vertical axis }
10246: procedure TeX_PlotGrid(Grid : TGrid); external 'dmath'; { Plots a grid on the graph }
10247: procedure TeX_WriteGraphTitle; external 'dmath'; Writes the title of the graph }
10248: procedure TeX_SetMaxCurv(NCurv : Byte); external 'dmath';
10249: { Sets the maximum number of curves and re-initializes their parameters }
10250: procedure TeX_SetPointParam(CurvIndex, Symbol, Size : Integer); external 'dmath';
10251: { Sets the point parameters for curve # CurvIndex }
10252: procedure TeX_SetLineParam(CurvIndex, Style : Integer;
10253:                             Width : Float; Smooth : Boolean); external 'dmath';
10254: { Sets the line parameters for curve # CurvIndex }
10255: procedure TeX_SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10256: { Sets the legend for curve # CurvIndex }
10257: procedure TeX_SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10258: { Sets the step for curve # CurvIndex }
10259: procedure TeX_PlotCurve(X, Y : TVector; Lb, Ub, CurvIndex : Integer); external 'dmath';
10260: { Plots a curve }
10261: procedure TeX_PlotCurveWithErrorBars(X, Y, S : TVector;
10262:                                       Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10263: { Plots a curve with error bars }
10264: procedure TeX_PlotFunc(Func : TFunc; X1, X2 : Float;
10265:                         Npt : Integer; CurvIndex : Integer); external 'dmath';
10266: { Plots a function }
10267: procedure TeX_WriteLegend(NCurv : Integer; ShowPoints, ShowLines : Boolean); external 'dmath';
10268: { Writes the legends for the plotted curves }
10269: procedure TeX_ConRec(Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix); external 'dmath';
10270: { Contour plot }
10271: function Xcm(X : Float) : Float; external 'dmath'; { Converts user coordinate X to cm }
10272: function Ycm(Y : Float) : Float; external 'dmath'; { Converts user coordinate Y to cm }
10273:
10274: //*****unit uPSI_SynPdf;
10275: Function RawUTF8ToPDFString( const Value : RawUTF8) : PDFString
10276: _DateTimeToPdfDate( ADate : TDateTime) : TPdfDate
10277: Function _PdfDateToDateTime( const AText : TPdfDate) : TDateTime
10278: Function PdfRect( Left, Top, Right, Bottom : Single) : TPdfRect;
10279: Function PdfRect1( const Box : TPdfBox) : TPdfRect;
10280: Function PdfBox( Left, Top, Width, Height : Single) : TPdfBox
10281: //Function _GetCharCount( Text : PAnsiChar) : integer
10282: //Procedure L2R( W : PWideChar; L : integer)
10283: Function PdfCoord( MM : single) : integer
10284: Function CurrentPrinterPaperSize : TPDPaperSize
10285: Function CurrentPrinterRes : TPoint
10286: Procedure GDICommentBookmark( MetaHandle : HDC; const aBookmarkName : RawUTF8)
10287: Procedure GDICommentOutline( MetaHandle : HDC; const aTitle : RawUTF8; aLevel : Integer)
10288: Procedure GDICommentLink( MetaHandle:HDC; const aBookmarkName:RawUTF8; const aRect : TRect)
10289: Const('Usp10','String').SetString( 'usp10.dll
10290: AddTypeS('TScriptState_enum', '( r0, r1, r2, r3, r4, fOverrideDirection, fInhibitSymSwap,
10291: 'fCharShape, fDigitSubstitute,fInhibitLigate,fDisplayZWG, fArabicNumContext, fGcpClusters )
10292: AddTypeS('TScriptState_set', 'set of TScriptState_enum
10293: //*****
10294:
10295: procedure SIRegister_PMrand(CL: TPSPascalCompiler); //ParkMiller
10296: begin
10297:   Procedure PMrandomize( I : word)
10298:   Function PMrandom : longint
10299:   Function Rrand : extended
10300:   Function Irand( N : word) : word
10301:   Function Brand( P : extended) : boolean
10302:   Function Nrand : extended
10303: end;
10304:
10305: procedure SIRegister_Spring_Cryptography_Utills(CL: TPSPascalCompiler);
10306: begin
10307:   Function Endian( x : LongWord) : LongWord
10308:   Function Endian64( x : Int64) : Int64
10309:   Function spRol( x : LongWord; y : Byte) : LongWord
10310:   Function spRor( x : LongWord; y : Byte) : LongWord
10311:   Function Ror64( x : Int64; y : Byte) : Int64
10312: end;
10313:
10314: procedure SIRegister_MapReader(CL: TPSPascalCompiler);
10315: begin
10316:   Procedure ClearModules
10317:   Procedure ReadMapFile( Fname : string)
10318:   Function AddressInfo( Address : dword) : string
10319: end;
10320:

```

```

10321: procedure SIRegister_LibTar(CL: TPSPascalCompiler);
10322: begin
10323:   TTarPermission', '( tpReadByOwner, tpWriteByOwner, tpExecuteByOw'
10324:   + 'ner, tpReadByGroup, tpWriteByGroup, tpExecuteByGroup, tpReadByOther, tpWri'
10325:   + 'teByOther, tpExecuteByOther )
10326:   TTarPermissions', 'set of TTarPermission
10327:   TFileType', '( ftNormal, ftLink, ftSymbolicLink, ftCharacter, ft'
10328:   + 'Block, ftDirectory, ftFifo, ftContiguous, ftDumpDir, ftMultiVolume, ftVolumeHeader;
10329:   TTarMode', '( tmSetUid, tmSetGid, tmSaveText )
10330:   TTarModes', 'set of TTarMode
10331:   TTarDirRec', 'record Name : STRING; Size : INT64; DateTime : TDA'
10332:   + 'teTime; Permissions : TTarPermissions; FileType : TFileType; LinkName : ST'
10333:   + 'RING; UID : INTEGER; GID : INTEGER; UserName : STRING; GroupName : STRING;'
10334:   + 'ChecksumOK : BOOLEAN; Mode : TTarModes; Magic : STRING; MajorDevNo : INTE'
10335:   + 'GER; MinorDevNo : INTEGER; FilePos : INT64; end
10336:   SIRegister_TTarArchive(CL);
10337:   SIRegister_TTarWriter(CL);
10338:   Function PermissionString( Permissions : TTarPermissions) : STRING
10339:   Function ConvertFilename( Filename : STRING) : STRING
10340:   Function FileTimeGMT( FileName : STRING) : TDateTime;
10341:   Function FileTimeGMT1( SearchRec : TSearchRec) : TDateTime;
10342:   Procedure ClearDirRec( var DirRec : TTarDirRec)
10343: end;
10344:
10345:
10346: //*****unit uPSI_TlHelp32;
10347: procedure SIRegister_TlHelp32(CL: TPSPascalCompiler);
10348: begin
10349:   Const('MAX_MODULE_NAME32', 'LongInt').SetInt( 255);
10350:   Function CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD) : THandle
10351:   Const('TH32CS_SNAPHEAPLIST', 'LongWord').SetUInt( $00000001);
10352:   Const('TH32CS_SNAPPROCESS', 'LongWord').SetUInt( $00000002);
10353:   Const('TH32CS_SNAPTHREAD', 'LongWord').SetUInt( $00000004);
10354:   Const('TH32CS_SNAPMODULE', 'LongWord').SetUInt( $00000008);
10355:   Const('TH32CS_INHERIT', 'LongWord').SetUInt( $80000000);
10356:   tagHEAPLIST32', 'record dwSize:DWORD;th32ProcessID:DWORD;th32HeapID:DWORD;dwFlags:DWORD;end';
10357:   AddTypeS('HEAPLIST32', 'tagHEAPLIST32
10358:   AddTypeS('TheapList32', 'tagHEAPLIST32
10359:   Const('HF32_DEFAULT', 'LongInt').SetInt( 1);
10360:   Const('HF32_SHARED', 'LongInt').SetInt( 2);
10361:   Function Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10362:   Function Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10363:   AddTypeS('tagHEAPENTRY32', 'record dwSize : DWORD; hHandle : THandle; dwAd'
10364:   + 'dress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
10365:   + 'dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end
10366:   AddTypeS('HEAPENTRY32', 'tagHEAPENTRY32
10367:   AddTypeS('TheapEntry32', 'tagHEAPENTRY32
10368:   Const('LF32_FIXED', 'LongWord').SetUInt( $00000001);
10369:   Const('LF32_FREE', 'LongWord').SetUInt( $00000002);
10370:   Const('LF32_MOVEABLE', 'LongWord').SetUInt( $00000004);
10371:   Function Heap32First( var lphe : THeapEntry32; th32ProcessID, th32HeapID : DWORD) : BOOL
10372:   Function Heap32Next( var lphe : THeapEntry32) : BOOL
10373:   DWORD; var lpNumberOfBytesRead : DWORD) : BOOL
10374:   AddTypeS('tagTHREADENTRY32', 'record dwSize : DWORD; cntUsage : DWORD; th3'
10375:   + '2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt'
10376:   + 'aPri : Longint; dwFlags : DWORD; end
10377:   AddTypeS('THREADENTRY32', 'tagTHREADENTRY32
10378:   AddTypeS('TThreadEntry32', 'tagTHREADENTRY32
10379:   Function Thread32First( hSnapshot : THandle; var lpthe : TThreadEntry32) : BOOL
10380:   Function Thread32Next( hSnapshot : THandle; var lpthe : TThreadEntry32) : BOOL
10381: end;
10382:   Const('EW_RESTARTWINDOWS', 'LongWord').SetUInt( $0042);
10383:   Const('EW_REBOOTSYSTEM', 'LongWord').SetUInt( $0043);
10384:   Const('EW_EXITANDEXECAPP', 'LongWord').SetUInt( $0044);
10385:   Const('ENDSESSION_LOGOFF', 'LongWord').SetUInt( DWORD ( $80000000 ));
10386:   Const('EWX_LOGOFF', 'LongInt').SetInt( 0);
10387:   Const('EWX_SHUTDOWN', 'LongInt').SetInt( 1);
10388:   Const('EWX_REBOOT', 'LongInt').SetInt( 2);
10389:   Const('EWX_FORCE', 'LongInt').SetInt( 4);
10390:   Const('EWX_POWEROFF', 'LongInt').SetInt( 8);
10391:   Const('EWX_FORCEIFHUNG', 'LongWord').SetUInt( $10);
10392:   Function GET_APPCOMMAND_LPARAM( const lParam : LongInt) : Shortint
10393:   Function GET_DEVICE_LPARAM( const lParam : LongInt) : Word
10394:   Function GET_MOUSEORKEY_LPARAM( const lParam : LongInt) : Word
10395:   Function GET_FLAGS_LPARAM( const lParam : LongInt) : Word
10396:   Function GET_KEYSTATE_LPARAM( const lParam : LongInt) : Word
10397:   Function GetWindowWord( hWnd : HWND; nIndex : Integer) : Word
10398:   Function SetWindowWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10399:   Function GetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
10400:   Function SetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
10401:   Function GetClassWord( hWnd : HWND; nIndex : Integer) : Word
10402:   Function SetClassWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10403:   Function GetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
10404:   Function SetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
10405:   Function GetDesktopWindow : HWND
10406:   Function GetParent( hWnd : HWND) : HWND
10407:   Function SetParent( hWndChild, hWndNewParent : HWND) : HWND
10408:   Function GetTopWindow( hWnd : HWND) : HWND
10409:   Function GetNextWindow( hWnd : HWND; uCmd : UINT) : HWND

```



```

10410: Function GetWindow( hWnd : HWND; uCmd : UINT ) : HWND
10411: //Delphi DFM
10412: Function LoadDFMFile2Strings(const AFile:string; AStrings:TStrings; var WasText:boolean):integer
10413: Function SaveStrings2DFMFile( AStrings : TStrings; const AFile : string ) : integer
10414: procedure GetHighlighters(AOwner: TComponent; AHighlighters: TStringList; AppendToLst: boolean);
10415: function GetHighlightersFilter(AHighlighters: TStringList): string;
10416: function GetHighlighterFromFileExt(AHighlighters: TStringList;Extension: string):TSynCustomHighlighter;
10417: Function ShowOwnedPopups( hWnd : HWND; fShow : BOOL ) : BOOL
10418: Function OpenIcon( hWnd : HWND ) : BOOL
10419: Function CloseWindow( hWnd : HWND ) : BOOL
10420: Function MoveWindow( hWnd : HWND; X, Y, nWidth, nHeight : Integer; bRepaint : BOOL ) : BOOL
10421: Function SetWindowPos(hWnd: HWND;hWndInsertAfter:HWND; X,Y,cx,cy : Integer; uFlags : UINT ) : BOOL
10422: Function IsWindowVisible( hWnd : HWND ) : BOOL
10423: Function IsIconic( hWnd : HWND ) : BOOL
10424: Function AnyPopup : BOOL
10425: Function BringWindowToTop( hWnd : HWND ) : BOOL
10426: Function IsZoomed( hWnd : HWND ) : BOOL
10427: Function IsWindow( hWnd : HWND ) : BOOL
10428: Function IsMenu( hMenu : HMENU ) : BOOL
10429: Function IsChild( hWndParent, hWnd : HWND ) : BOOL
10430: Function DestroyWindow( hWnd : HWND ) : BOOL
10431: Function ShowWindow( hWnd : HWND; nCmdShow : Integer ) : BOOL
10432: Function AnimateWindow( hWnd : HWND; dwTime : DWORD; dwFlags : DWORD ) : BOOL
10433: Function ShowWindowAsync( hWnd : HWND; nCmdShow : Integer ) : BOOL
10434: Function FlashWindow( hWnd : HWND; bInvert : BOOL ) : BOOL
10435: Function IsWindowUnicode( hWnd : HWND ) : BOOL
10436: Function EnableWindow( hWnd : HWND; bEnable : BOOL ) : BOOL
10437: Function IsWindowEnabled( hWnd : HWND ) : BOOL
10438:
10439: procedure SIRegister_IDECmdLine(CL: TPSPascalCompiler);
10440: begin
10441:   const ('ShowSetupDialogOptLong','String').SetString( '--setup
10442:   PrimaryConfPathOptLong','String').SetString( '--primary-config-path=
10443:   PrimaryConfPathOptShort','String').SetString( '--pcp=
10444:   SecondaryConfPathOptLong','String').SetString( '--secondary-config-path=
10445:   SecondaryConfPathOptShort','String').SetString( '--scp=
10446:   NoSplashScreenOptLong','String').SetString( '--no-splash-screen
10447:   NoSplashScreenOptShort','String').SetString( '--nsc
10448:   StartedByStartLazarusOpt','String').SetString( '--started-by-startlazarus
10449:   SkipLastProjectOpt','String').SetString( '--skip-last-project
10450:   DebugLogOpt','String').SetString( '--debug-log=
10451:   DebugLogOptEnable','String').SetString( '--debug-enable=
10452:   LanguageOpt','String').SetString( '--language=
10453:   LazarusDirOpt','String').SetString( '--lazarusdir=
10454: Procedure ParseCommandLine(aCmdLineParams:TStrings;out IDEPid:Int;out ShowSplashScreen:boolean);
10455: Function GetCommandLineParameters( aCmdLineParams : TStrings; isStartLazarus:Boolean) : string
10456: Function ExtractPrimaryConfigPath( aCmdLineParams : TStrings ) : string
10457: Function IsHelpRequested : Boolean
10458: Function IsVersionRequested : boolean
10459: Function GetLanguageSpecified : string
10460: Function ParamIsOption( ParamIndex : integer; const Option : string ) : boolean
10461: Function ParamIsOptionPlusValue(ParamIndex:integer;const Option:string;out AValue:string):bool;
10462: Procedure ParseNoGuiCmdLineParams
10463: Function ExtractCmdLineFileNames : TStrings
10464: end;
10465:
10466:
10467: procedure SIRegister_LazFileUtils(CL: TPSPascalCompiler);
10468: begin
10469:   Function CompareFileNames( const Filename1, Filename2 : string ) : integer
10470:   Function CompareFileNamesIgnoreCase( const Filename1, Filename2 : string ) : integer
10471:   Function CompareFileExt( const Filename, Ext : string; CaseSensitive : boolean ) : integer;
10472:   Function CompareFileExt1( const Filename, Ext : string ) : integer;
10473:   Function CompareFilenameStarts( const Filename1, Filename2 : string ) : integer
10474:   Function CompareFileNames(Filename1:PChar;Len1:integer; Filename2:PChar;Len2:integer):integer
10475:   Function CompareFileNamesP( Filename1, Filename2 : PChar; IgnoreCase : boolean ) : integer
10476:   Function DirPathExists( DirectoryName : string ) : boolean
10477:   Function DirectoryIsWritable( const DirectoryName : string ) : boolean
10478:   Function ExtractFileNameOnly( const AFilename : string ) : string
10479:   Function FilenameIsAbsolute( const TheFilename : string ) : boolean
10480:   Function FilenameIsWinAbsolute( const TheFilename : string ) : boolean
10481:   Function FilenameIsUnixAbsolute( const TheFilename : string ) : boolean
10482:   Function ForceDirectory( DirectoryName : string ) : boolean
10483: Procedure CheckIfFileIsExecutable( const AFilename : string)
10484: Procedure CheckIfFileIsSymlink( const AFilename : string)
10485: Function FileIsText( const AFilename : string ) : boolean
10486: Function FileIsText2( const AFilename : string; out FileReadable : boolean ) : boolean
10487: Function FilenameIsTrimmed( const TheFilename : string ) : boolean
10488: Function FilenameIsTrimmed2( StartPos : PChar; NameLen : integer ) : boolean
10489: Function TrimFilename( const AFilename : string ) : string
10490: Function ResolveDots( const AFilename : string ) : string
10491: Procedure ForcePathDelims( var FileName : string)
10492: Function GetForcedPathDelims( const FileName : string ) : String
10493: Function CleanAndExpandFilename( const Filename : string ) : string
10494: Function CleanAndExpandDirectory( const Filename : string ) : string
10495: Function TrimAndExpandFilename( const Filename : string; const BaseDir : string ) : string
10496: Function TrimAndExpandDirectory( const Filename : string; const BaseDir : string ) : string
10497: Function TryCreateRelativePath(const Dest,Source:String; UsePointDirectory:bool;
AlwaysRequireSharedBaseFolder : Boolean; out RelPath : String ) : Boolean

```

```

10498: Function CreateRelativePath( const Filename,BaseDirectory:string; UsePointDirectory:boolean;
AlwaysRequireSharedBaseFolder: Boolean) : string
10499: Function FileIsInPath( const Filename, Path : string) : boolean
10500: Function AppendPathDelim( const Path : string) : string
10501: Function ChompPathDelim( const Path : string) : string
10502: Function CreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
10503: Function CreateRelativeSearchPath( const SearchPath, BaseDirectory : string) : string
10504: Function MinimizeSearchPath( const SearchPath : string) : string
10505: Function FindPathInSearchPath(APath:PChar;APathLen:int;SearchPath:PChar;SearchPathLen:int):PChar;
10506: (*Function FileExistsUTF8( const Filename : string) : boolean
10507: Function FileAgeUTF8( const FileName : string) : Longint
10508: Function DirectoryExistsUTF8( const Directory : string) : Boolean
10509: Function ExpandFileNameUTF8( const FileName : string; BaseDir : string) : string
10510: Function FindFirstUTF8(const Path:string; Attr: Longint; out Rslt : TSearchRec) : Longint
10511: Function FindNextUTF8( var Rslt : TSearchRec) : Longint
10512: Procedure FindCloseUTF8( var F : TSearchRec)
10513: Function FileSetDateUTF8( const FileName : string; Age : Longint) : Longint
10514: Function FileGetAttrUTF8( const FileName : string) : Longint
10515: Function FileSetAttrUTF8( const FileName : string; Attr : longint) : Longint
10516: Function DeleteFileUTF8( const FileName : string) : Boolean
10517: Function RenameFileUTF8( const OldName, NewName : string) : Boolean
10518: Function FileSearchUTF8( const Name, DirList : string; ImplicitCurrentDir : Boolean) : string
10519: Function FileIsReadOnlyUTF8( const FileName : string) : Boolean
10520: Function GetCurrentDirUTF8 : string
10521: Function SetCurrentDirUTF8( const NewDir : string) : Boolean
10522: Function CreateDirUTF8( const NewDir : string) : Boolean
10523: Function RemoveDirUTF8( const Dir : string) : Boolean
10524: Function ForceDirectoriesUTF8( const Dir : string) : Boolean
10525: Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
10526: Function FileCreateUTF8( const FileName : string) : THandle;
10527: Function FileCreateUTF8( const FileName : string; Rights : Cardinal) : THandle;
10528: Function FileCreateUtf8( const FileName : string; ShareMode : Integer; Rights : Cardinal) : THandle;
10529: Function FileSizeUtf8( const Filename : string) : int64
10530: Function GetFileDescription( const AFilename : string) : string
10531: Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
10532: Function GetAppConfigFileUTF8( Global : Boolean; SubDir : boolean; CreateDir : boolean) : string
10533: Function GetTempFileNameUTF8( const Dir, Prefix : string) : string*)
10534: Function IsUNCPath( const Path : string) : boolean
10535: Function ExtractUNCVolume( const Path : string) : string
10536: Function ExtractFileRoot( FileName : string) : string
10537: Function GetDarwinSystemFilename( FileName : string) : string
10538: Procedure SplitCmdLineParams( const Params : string; ParamList : TStrings; ReadBackslash : boolean)
10539: Function StrToCmdLineParam( const Param : string) : string
10540: Function MergeCmdLineParams( ParamList : TStrings) : string
10541: Procedure InvalidateFileStateCache( const Filename : string)
10542: Function FindAllFiles(const SearchPath:String;SearchMask:String;SearchSubDirs:Boolean):TStringList);
10543: Function FindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
10544: Function ReadFileToString( const Filename : string) : string
10545: type
10546:   TCopyFileFlag = ( cffOverwriteFile,
10547:     cffCreateDestDirectory, cffPreserveTime );
10548:   TCopyFileFlags = set of TCopyFileFlag;*)
10549:   TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime)
10550:   TCopyFileFlags', 'set of TCopyFileFlag
10551:   Function CopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
10552: end;
10553:
10554: procedure SIRegister_lazMasks(CL: TPSPascalCompiler);
10555: begin
10556:   TMaskCharType', '( mcChar, mcCharSet, mcAnyChar, mcAnyText )
10557:   SIRegister_TMask(CL);
10558:   SIRegister_TParseStringList(CL);
10559:   SIRegister_TMaskList(CL);
10560:   Function MatchesMask( const FileName, Mask : string; const CaseSensitive : Boolean) : Boolean
10561:   Function MatchesWindowsMask( const FileName, Mask : string; const CaseSensitive : Boolean) : Bool;
10562:   Function MatchesMaskList(const FileName,Mask:String;Separator:Char;const CaseSensitive:Boolean):Bool;
10563:   Function MatchesWindowsMaskList(const FileName,Mask:String;Separat:Char;const CaseSensitive:Bool):Bool;
10564: end;
10565:
10566: procedure SIRegister_JvShellHook(CL: TPSPascalCompiler);
10567: begin
10568:   //PShellHookInfo', '^TShellHookInfo // will not work
10569:   TShellHookInfo', 'record hwnd : THandle; rc : TRect; end
10570:   SHELLHOOKINFO', 'TShellHookInfo
10571:   LPSHELLHOOKINFO', 'PShellHookInfo
10572:   TJvShellHookEvent', 'Procedure ( Sender : TObject; var Msg : TMessage)
10573:   SIRegister_TJvShellHook(CL);
10574:   Function InitJvShellHooks : Boolean
10575:   Procedure UnInitJvShellHooks
10576: end;
10577:
10578: procedure SIRegister_JvExControls(CL: TPSPascalCompiler);
10579: begin
10580:   TDlgCode', '( dcWantAllKeys, dcWantArrows, dcWantChars, dcButton'
10581:   +', dcHasSetSel, dcWantTab, dcNative )
10582:   TDlgCodes', 'set of TDlgCode
10583:   'dcWantMessage','').SetString( dcWantAllKeys);
10584:   SIRegister_IJvExControl(CL);
10585:   SIRegister_IJvDenySubClassing(CL);

```

```

10586:   SIRegister_TStructPtrMessage(CL);
10587: Procedure SetDotNetFrameColors( FocusedColor, UnfocusedColor : TColor);
10588: Procedure DrawDotNetControl( Control : TWinControl; AColor : TColor; InControl : Boolean);
10589: Procedure DrawDotNetControl1( DC : HDC; R : TRect; AColor : TColor; UseFocusedColor : Boolean);
10590: Procedure HandleDotNetHighlighting( Control:TWinControl;const Msg:TMessage;MouseOver:Boolean;Color:TColor);
10591: Function CreateWMMessage( Msg : Integer; WParam : Integer; LParam : Longint) : TMessage;
10592: Function CreateWMMessage1( Msg : Integer; WParam : Integer; LParam : TControl) : TMessage;
10593: Function SmallPointToLong( const Pt : TSmallPoint) : Longint
10594: Function ShiftStateToKeyData( Shift : TShiftState) : Longint
10595: Function GetFocusedControl( AControl : TControl) : TWinControl
10596: Function DlgcToDlgCodes( Value : Longint) : TDlgCodes
10597: Function DlgCodesToDlgc( Value : TDlgCodes) : Longint
10598: Procedure GetHintColor( var HintInfo : THintInfo; AControl : TControl; HintColor : TColor)
10599: Function DispatchIsDesignMsg( Control : TControl; var Msg : TMessage) : Boolean
10600:   SIRegister_TJvExControl(CL);
10601:   SIRegister_TJvExWinControl(CL);
10602:   SIRegister_TJvExCustomControl(CL);
10603:   SIRegister_TJvExGraphicControl(CL);
10604:   SIRegister_TJvExHintWindow(CL);
10605:   SIRegister_TJvExPubGraphicControl(CL);
10606: end;
10607:
10608: (*-----*)
10609: procedure SIRegister_EncdDecd(CL: TPSPascalCompiler);
10610: begin
10611:   Procedure EncodeStream( Input, Output : TStream)
10612:   Procedure DecodeStream( Input, Output : TStream)
10613:   Function EncodeString1( const Input : string) : string
10614:   Function DecodeString1( const Input : string) : string
10615: end;
10616:
10617: (*-----*)
10618: procedure SIRegister_SockAppReg(CL: TPSPascalCompiler);
10619: begin
10620:   SIRegister_TWebAppRegInfo(CL);
10621:   SIRegister_TWebAppRegList(CL);
10622:   Procedure GetRegisteredWebApps( AList : TWebAppRegList)
10623:   Procedure RegisterWebApp( const AFileName, AProgID : string)
10624:   Procedure UnregisterWebApp( const AProgID : string)
10625:   Function FindRegisteredWebApp( const AProgID : string) : string
10626:   Function CreateRegistry( InitializeNewFile : Boolean) : TCustomIniFile
10627:   'sUDPPort', 'String').SetString( 'UDPPort
10628: end;
10629:
10630: procedure SIRegister_PJEnvVars(CL: TPSPascalCompiler);
10631: begin
10632:   // TStringDynArray, 'array of string
10633:   Function GetEnvVarValue( const VarName : string) : string
10634:   Function SetEnvVarValue( const VarName, VarValue : string) : Integer
10635:   Function DeleteEnvVar( const VarName : string) : Integer
10636:   Function CreateEnvBlock(const NewEnv:TStrings;const IncludeCurrent:Bool;const Buffer:string;const
BufSize:Int):Int;
10637:   Function ExpandEnvVars( const Str : string) : string
10638:   Function GetAllEnvVars( const Vars : TStrings) : Integer
10639:   Procedure GetAllEnvVarNames( const Names : TStrings);
10640:   Function GetAllEnvVarNames1 : TStringDynArray;
10641:   Function EnvBlockSize : Integer
10642:   TPJEnvVarsEnum, 'Procedure ( const VarName : string; Data : TObject)
10643:   SIRegister_TPJEnvVarsEnumerator(CL);
10644:   SIRegister_TPJEnvVars(CL);
10645:   FindClass('TOBJECT'),'EPJEnvVars
10646:   FindClass('TOBJECT'),'EPJEnvVars
10647:   //Procedure Register
10648: end;
10649:
10650: (*-----*)
10651: procedure SIRegister_PJConsoleApp(CL: TPSPascalCompiler);
10652: begin
10653:   'cOneSecInMS','LongInt').SetInt( 1000);
10654:   //'cDefTimeSlice','LongInt').SetInt( 50);
10655:   //'cDefMaxExecTime','').SetString( cOneMinInMS);
10656:   'cAppErrorMask','LongInt').SetInt( 1 shl 29);
10657:   Function IsApplicationError( const ErrCode : LongWord) : Boolean
10658:   TPJConsoleAppPriority, '( cpDefault, cpHigh, cpNormal, cpIdle, cpRealTime )
10659:   TPJConsoleColors, 'record Foreground : TPJConsoleColor; Background:TPJConsoleColor; end;
10660:   Function MakeConsoleColors( const AForeground, ABackground : TPJConsoleColor):TPJConsoleColors;
10661:   Function MakeConsoleColors1( const AForeground, ABackground : TColor) : TPJConsoleColors;
10662:   Function MakeConsoleColors2( const AForeground, ABackground : TAlphaColor) : TPJConsoleColors;
10663:   Function MakeSize( const ACX, ACY : LongInt) : TSize
10664:   SIRegister_TPJCustomConsoleApp(CL);
10665:   SIRegister_TPJConsoleApp(CL);
10666: end;
10667:
10668: procedure SIRegister_ip_misc(CL: TPSPascalCompiler);
10669: begin
10670:   INVALID_IP_ADDRESS, 'LongWord').SetUInt( $ffffff);
10671:   t_encoding, '( uencode, base64, mime )
10672:   Function internet_date( date : TDateTime) : string
10673:   Function lookup_hostname( const hostname : string) : longint

```

```

10674: Function my_hostname : string
10675: Function my_ip_address : longint
10676: Function ip2string( ip_address : longint) : string
10677: Function resolve_hostname( ip : longint) : string
10678: Function address_from( const s : string; count : integer) : string
10679: Function encode_base64( data : TStream) : TStringList
10680: Function decode_base64( source : TStringList) : TMemoryStream
10681: Function posn( const s, t : string; count : integer) : integer
10682: Function poscn( c : char; const s : string; n : integer) : integer
10683: Function filename_of( const s : string) : string
10684: //Function trim( const s : string) : string
10685: //Procedure setlength( var s : string; l : byte)
10686: Function TimeZoneBias : longint
10687: Function eight2seven_quoteprint( const s : string) : string
10688: Function eight2seven_german( const s : string) : string
10689: Function seven2eight_quoteprint( const s : string) : string end;
10690: type in_addr, record s_bytes : array[1..4] of byte; end;
10691: Function socketerror : cint
10692: Function fpsocket( domain : cint; xtype : cint; protocol : cint) : cint
10693: Function fprecv( s : cint; buf : __pointer; len : size_t; flags : cint) : ssize_t
10694: Function fpsend( s : cint; msg : __pointer; len : size_t; flags : cint) : ssize_t
10695: //Function fbind( s : cint; addrx : psockaddr; addrlen : tsocklen) : cint
10696: Function fplisten( s : cint; backlog : cint) : cint
10697: //Function fpaccept( s : cint; addrx : psockaddr; addrlen : plongint) : cint
10698: //Function fpconnect( s : cint; name : psockaddr; namelen : tsocklen) : cint
10699: //Function fpgetsockname( s : cint; name : psockaddr; namelen : psocklen) : cint
10700: Function NetAddrToStr( Entry : in_addr) : String
10701: Function HostAddrToStr( Entry : in_addr) : String
10702: Function StrToHostAddr( IP : String) : in_addr
10703: Function StrToNetAddr( IP : String) : in_addr
10704: SOL_SOCKET, 'LongWord').SetUInt( $ffff);
10705: cint8, 'shortint
10706: cuint8, 'byte
10707: cchar, 'cint8
10708: cschar, 'cint8
10709: cuchar, 'cuint8
10710: cint16, 'smallint
10711: cuint16, 'word
10712: cshort, 'cint16
10713: cushort, 'cint16
10714: cushort, 'cuint16
10715: cint32, 'longint
10716: cuint32, 'longword
10717: cint, 'cint32
10718: csint, 'cint32
10719: cuint, 'cuint32
10720: csigned, 'cint
10721: cunsigned, 'cuint
10722: cint64, 'int64
10723: clonglong, 'cint64
10724: cslonglong, 'cint64
10725: cbool, 'longbool
10726: cfloat, 'single
10727: cdouble, 'double
10728: clongdouble, 'extended
10729:
10730: procedure SIRegister_uLkJSON(CL: TPSPascalCompiler);
10731: begin
10732:   TlkJSONtypes', '(jsBase,jsNumber,jsString,jsBoolean,jsNull,jsList,jsObject )
10733:   SIRegister_TlkJSONdotnetclass(CL);
10734:   SIRegister_TlkJSONbase(CL);
10735:   SIRegister_TlkJSONnumber(CL);
10736:   SIRegister_TlkJSONstring(CL);
10737:   SIRegister_TlkJSONboolean(CL);
10738:   SIRegister_TlkJSONnull(CL);
10739:   TlkJSONFuncEnum', 'Procedure ( ElName : string; Elem : TlkJSONba'
10740:   +'se; data : TObject; var Continue : Boolean)
10741:   SIRegister_TlkJSONcustomlist(CL);
10742:   SIRegister_TlkJSONlist(CL);
10743:   SIRegister_TlkJSONobjectmethod(CL);
10744:   TlkHashItem', 'record hash : cardinal; index : Integer; end
10745:   TlkHashFunction', 'Function ( const ws : WideString) : cardinal
10746:   SIRegister_TlkHashTable(CL);
10747:   SIRegister_TlkBalTree(CL);
10748:   SIRegister_TlkJSONobject(CL);
10749:   SIRegister_TlkJSON(CL);
10750:   SIRegister_TlkJSONstreamed(CL);
10751: Function GenerateReadableText( vObj : TlkJSONbase; var vLevel : Integer) : string
10752: end;
10753:
10754: procedure SIRegister_ZSysUtils(CL: TPSPascalCompiler);
10755: begin
10756:   TZListSortCompare', 'Function (Item1, Item2 : TObject): Integer
10757:   SIRegister_TZSortedList(CL);
10758:   Function zFirstDelimiter( const Delimiters, Str : string) : Integer
10759:   Function zLastDelimiter( const Delimiters, Str : string) : Integer
10760:   //Function MemLCompUnicode( P1, P2 : PWideChar; Len : Integer) : Boolean
10761:   //Function MemLCompAnsi( P1, P2 : PAnsiChar; Len : Integer) : Boolean
10762:   Function zStartsWith( const Str, SubStr : WideString) : Boolean;

```



```

10763: Function StartsWith1( const Str, SubStr : RawByteString ) : Boolean;
10764: Function EndsWith( const Str, SubStr : WideString ) : Boolean;
10765: Function EndsWith1( const Str, SubStr : RawByteString ) : Boolean;
10766: Function SQLStrToFloatDef( Str : RawByteString; Def : Extended ) : Extended;
10767: Function SQLStrToFloatDef1( Str : String; Def : Extended ) : Extended;
10768: Function SQLStrToFloat( const Str : AnsiString ) : Extended
10769: //Function BufferToStr( Buffer : PWideChar; Length : LongInt ) : string;
10770: //Function BufferToStr1( Buffer : PAnsiChar; Length : LongInt ) : string;
10771: Function BufferToBytes( Buffer : TObject; Length : LongInt ) : TByteDynArray
10772: Function StrToBoolEx( Str : string ) : Boolean
10773: Function BoolToStrEx( Bool : Boolean ) : String
10774: Function IsIpAddr( const Str : string ) : Boolean
10775: Function zSplitString( const Str, Delimiters : string ) : TStrings
10776: Procedure PutSplitString( List : TStrings; const Str, Delimiters : string )
10777: Procedure AppendSplitString( List : TStrings; const Str, Delimiters : string )
10778: Function ComposeString( List : TStrings; const Delimiter : string ) : string
10779: Function FloatToSQLStr( Value : Extended ) : string
10780: Procedure PutSplitStringEx( List : TStrings; const Str, Delimiter : string )
10781: Function SplitStringEx( const Str, Delimiter : string ) : TStrings
10782: Procedure AppendSplitStringEx( List : TStrings; const Str, Delimiter : string )
10783: Function zBytesToStr( const Value : TByteDynArray ) : AnsiString
10784: Function zStrToBytes( const Value : AnsiString ) : TByteDynArray;
10785: Function StrToBytes1( const Value : UTF8String ) : TByteDynArray;
10786: Function StrToBytes2( const Value : RawByteString ) : TByteDynArray;
10787: Function StrToBytes3( const Value : WideString ) : TByteDynArray;
10788: Function StrToBytes4( const Value : UnicodeString ) : TByteDynArray;
10789: Function BytesToVar( const Value : TByteDynArray ) : Variant
10790: Function VarToBytes( const Value : Variant ) : TByteDynArray
10791: Function AnsiSQLDateToDateTime( const Value : string ) : TDateTime
10792: Function TimestampStrToDateTime( const Value : string ) : TDateTime
10793: Function DateTimeToAnsiSQLDate( Value : TDateTime; WithMMSec : Boolean ) : string
10794: Function EncodeCString( const Value : string ) : string
10795: Function DecodeCString( const Value : string ) : string
10796: Function zReplaceChar( const Source, Target : Char; const Str : string ) : string
10797: Function MemPas( Buffer : PChar; Length : LongInt ) : string
10798: Procedure DecodeSQLVersioning(const FullVersion:Integer;out MajorVersion:Integer;out MinorVersion:Integer;out
SubVersion:Integer);
10799: Function EncodeSQLVersioning(const MajorVersion:Integer;const MinorVersion:Integer;const
SubVersion:Integer):Integer;
10800: Function FormatSQLVersion( const SQLVersion : Integer ) : String
10801: Function ZStrToFloat( Value : AnsiChar ) : Extended;
10802: Function ZStrToFloat1( Value : AnsiString ) : Extended;
10803: Procedure ZSetString( const Src : AnsiChar; var Dest : AnsiString);
10804: Procedure ZSetString1( const Src : AnsiChar; const Len : Cardinal; var Dest : AnsiString);
10805: Procedure ZSetString2( const Src : AnsiChar; var Dest : UTF8String);
10806: Procedure ZSetString3( const Src : AnsiChar; const Len : Cardinal; var Dest : UTF8String);
10807: Procedure ZSetString4( const Src : AnsiChar; const Len : Cardinal; var Dest : WideString);
10808: Procedure ZSetString5( const Src : AnsiChar; var Dest : RawByteString);
10809: Procedure ZSetString6( const Src : AnsiChar; const Len : Cardinal; var Dest : RawByteString);
10810: end;
10811:
10812: unit uPSI_ZEncoding;
10813: Function StringToAnsiEx( const s : String; const FromCP, ToCP : Word ) : RawByteString
10814: Function AnsiToStringEx( const s : RawByteString; const FromCP, ToCP : Word ) : String
10815: Function ZRawToUnicode( const S : RawByteString; const CP : Word ) : WideString
10816: Function ZUnicodeToRaw( const US : WideString; CP : Word ) : RawByteString
10817: Function ZConvertAnsiToRaw( const Src : AnsiString; const RawCP : Word ) : RawByteString
10818: Function ZConvertRawToAnsi( const Src : RawByteString; const RawCP : Word ) : AnsiString
10819: Function ZConvertAnsiToUTF8( const Src : AnsiString ) : UTF8String
10820: Function ZConvertUTF8ToAnsi( const Src : UTF8String ) : AnsiString
10821: Function ZConvertRawToUTF8( const Src : RawByteString; const CP : Word ) : UTF8String
10822: Function ZConvertUTF8ToRaw( const Src : UTF8String; const CP : Word ) : RawByteString
10823: Function ZConvertRawToString( const Src : RawByteString; const RawCP, StringCP : Word ) : String
10824: Function ZConvertStringToRaw( const Src : String; const StringCP, RawCP : Word ) : RawByteString
10825: Function ZConvertStringToRawWithAutoEncode(const Src:String;const StringCP,RawCP:Word):RawByteString;
10826: Function ZConvertUTF8ToString( const Src : UTF8String; const StringCP : Word ) : String
10827: Function ZConvertStringToUTF8( const Src : String; const StringCP : Word ) : UTF8String
10828: Function ZConvertStringToUTF8WithAutoEncode( const Src : String; const StringCP : Word ) : UTF8String
10829: Function ZConvertStringToAnsi( const Src : String; const StringCP : Word ) : AnsiString
10830: Function ZConvertStringToAnsiWithAutoEncode( const Src : String; const StringCP : Word ) : AnsiString
10831: Function ZConvertAnsiToString( const Src : AnsiString; const StringCP : Word ) : String
10832: Function ZConvertUnicodeToString( const Src : WideString; const StringCP : Word ) : String
10833: Function ZConvertUnicodeToString_CPUTF8( const Src : WideString; const StringCP : Word ) : String
10834: Function ZConvertStringToUnicode( const Src : String; const StringCP : Word ) : WideString
10835: Function ZConvertString_CPUTF8ToUnicode( const Src : String; const StringCP : Word ) : WideString
10836: Function ZConvertStringToUnicodeWithAutoEncode( const Src : String; const StringCP : Word ) : WideString
10837: Function ZMoveAnsiToRaw( const Src : AnsiString; const RawCP : Word ) : RawByteString
10838: Function ZMoveRawToAnsi( const Src : RawByteString; const RawCP : Word ) : AnsiString
10839: Function ZMoveAnsiToUTF8( const Src : AnsiString ) : UTF8String
10840: Function ZMoveUTF8ToAnsi( const Src : UTF8String ) : AnsiString
10841: Function ZMoveRawToUTF8( const Src : RawByteString; const CP : Word ) : UTF8String
10842: Function ZMoveUTF8ToRaw( const Src : UTF8String; const CP : Word ) : RawByteString
10843: Function ZMoveStringToAnsi( const Src : String; const StringCP : Word ) : AnsiString
10844: Function ZMoveAnsiToString( const Src : AnsiString; const StringCP : Word ) : String
10845: Function ZMoveRawToString( const Src : RawByteString; const RawCP, StringCP : Word ) : String
10846: Function ZMoveStringToRaw( const Src : String; const StringCP, RawCP : Word ) : RawByteString
10847: Function ZMoveUTF8ToString( const Src : UTF8String; StringCP : Word ) : String
10848: Function ZMoveStringToUTF8( const Src : String; const StringCP : Word ) : UTF8String
10849: Function ZUnknownRawToUnicode( const S : RawByteString; const CP : Word ) : WideString

```

```

10850: Function ZUnknownRawToUnicodeWithAutoEncode( const S : RawByteString; const CP : Word) : WideString
10851: Function ZUnicodeToUnknownRaw( const US : WideString; CP : Word) : RawByteString
10852: Function ZDefaultSystemCodePage : Word
10853: Function ZCompatibleCodePages( const CP1, CP2 : Word) : Boolean
10854:
10855:
10856: procedure SIRegister_BoldComUtils(CL: TPSPascalCompiler);
10857: begin
10858:   'RPC_C_AUTHN_LEVEL_DEFAULT', 'LongInt').SetInt( 0);
10859:   ('RPC_C_AUTHN_LEVEL_NONE', 'LongInt').SetInt( 1);
10860:   ('RPC_C_AUTHN_LEVEL_CONNECT', 'LongInt').SetInt( 2);
10861:   ('RPC_C_AUTHN_LEVEL_CALL', 'LongInt').SetInt( 3);
10862:   ('RPC_C_AUTHN_LEVEL_PKT', 'LongInt').SetInt( 4);
10863:   ('RPC_C_AUTHN_LEVEL_PKT_INTEGRITY', 'LongInt').SetInt( 5);
10864:   ('RPC_C_AUTHN_LEVEL_PKT_PRIVACY', 'LongInt').SetInt( 6);
10865:   (('alDefault', '1').SetString( RPC_C_AUTHN_LEVEL_DEFAULT);
10866:   ('alNone', '2').SetString( RPC_C_AUTHN_LEVEL_NONE);
10867:   ('alConnect', '3').SetString( RPC_C_AUTHN_LEVEL_CONNECT);
10868:   ('alCall', '4').SetString( RPC_C_AUTHN_LEVEL_CALL);
10869:   ('alPacket', '5').SetString( RPC_C_AUTHN_LEVEL_PKT);
10870:   ('alPacketIntegrity', '6').SetString( RPC_C_AUTHN_LEVEL_PKT_INTEGRITY);
10871:   ('alPacketPrivacy', '7').SetString( RPC_C_AUTHN_LEVEL_PKT_PRIVACY);}
10872:   ('RPC_C_IMP_LEVEL_DEFAULT', 'LongInt').SetInt( 0);
10873:   ('RPC_C_IMP_LEVEL_ANONYMOUS', 'LongInt').SetInt( 1);
10874:   ('RPC_C_IMP_LEVEL_IDENTIFY', 'LongInt').SetInt( 2);
10875:   ('RPC_C_IMP_LEVEL_IMPERSONATE', 'LongInt').SetInt( 3);
10876:   ('RPC_C_IMP_LEVEL_DELEGATE', 'LongInt').SetInt( 4);
10877:   (('ilDefault', '0').SetString( RPC_C_IMP_LEVEL_DEFAULT);
10878:   ('ilAnonymous', '1').SetString( RPC_C_IMP_LEVEL_ANONYMOUS);
10879:   ('ilIdentify', '2').SetString( RPC_C_IMP_LEVEL_IDENTIFY);
10880:   ('ilImpersonate', '3').SetString( RPC_C_IMP_LEVEL_IMPERSONATE);
10881:   ('ilDelegate', '4').SetString( RPC_C_IMP_LEVEL_DELEGATE);}
10882:   ('EOAC_NONE', 'LongWord').SetUInt( $0);
10883:   ('EOAC_DEFAULT', 'LongWord').SetUInt( $800);
10884:   ('EOAC_MUTUAL_AUTH', 'LongWord').SetUInt( $1);
10885:   ('EOAC_STATIC_CLOACKING', 'LongWord').SetUInt( $20);
10886:   ('EOAC_DYNAMIC_CLOACKING', 'LongWord').SetUInt( $40);
10887:   ('EOAC_ANY_AUTHORITY', 'LongWord').SetUInt( $80);
10888:   ('RPC_C_AUTHN_WINNT', 'LongInt').SetInt( 10);
10889:   ('RPC_C_AUTHN_NONE', 'LongInt').SetInt( 0);
10890:   ('RPC_C_AUTHN_NAME', 'LongInt').SetInt( 1);
10891:   ('RPC_C_AUTHN_DCE', 'LongInt').SetInt( 2);
10892:   FindClass('TOBJECT'), 'EBoldCom
10893: Function BoldVariantIsType( V : OleVariant; TypeCode : Integer) : Boolean
10894: Function BoldMemoryToVariant( const Buffer, BufSize : Integer) : OleVariant
10895: Function BoldStreamToVariant( Stream : TStream) : OleVariant
10896: Function BoldStringsToVariant( Strings : TStrings) : OleVariant
10897: Function BoldVariantToMemory( V : OleVariant; var Buffer, BufSize : Integer) : Integer
10898: Function BoldVariantToStream( V : OleVariant; Stream : TStream) : Integer
10899: Function BoldVariantArrayOfArraysOfStringToStrings( V : OleVariant; Strings : TStrings) : Integer
10900: Function BoldVariantIsNamedValues( V : OleVariant) : Boolean
10901: Function BoldCreateNamedValues(const Names:array of string;const Values:array of OleVariant):OleVariant;
10902: Function BoldGetNamedValue( Data : OleVariant; const Name : string) : OleVariant
10903: Procedure BoldSetNamedValue( Data : OleVariant; const Name : string; Value : OleVariant)
10904: Function BoldCreateGUID : TGUID
10905: Function BoldCreateComObject( const ClsId, IID : TGUID; out Obj : variant; out Res : HRESULT) : Boolean
10906: Function BoldCreateRemoteComObject(const HostName:string;const ClsId,IID:TGUID;out Obj:variant;out
Res:HRESULT):Bool;
10907: Procedure BoldInitializeComSecurity( AuthenticationLevel, ImpersonationLevel : longint)
10908: Procedure BoldSetSecurityForInterface(AuthenticationLevel,ImpersonationLevel:longint;Unk:IUnknown);
10909: end;
10910:
10911: (*-----*)
10912: procedure SIRegister_BoldIsoDateTime(CL: TPSPascalCompiler);
10913: begin
10914:   Function ParseISODate( s : string) : TDateTime
10915:   Function ParseISODateTime( s : string) : TDateTime
10916:   Function ParseISOTime( str : string) : TDateTime
10917: end;
10918:
10919: (*-----*)
10920: procedure SIRegister_BoldGUIDUtils(CL: TPSPascalCompiler);
10921: begin
10922:   Function BoldCreateGUIDAsString( StripBrackets : Boolean) : string
10923:   Function BoldCreateGUIDWithBracketsAsString : string
10924: end;
10925:
10926: procedure SIRegister_BoldFileHandler(CL: TPSPascalCompiler);
10927: begin
10928:   FindClass('TOBJECT'), 'T-BoldFileHandler
10929:   FindClass('TOBJECT'), 'T-BoldDiskFileHandler
10930:   //T-BoldFileHandlerClass', 'class of T-BoldFileHandler
10931:   T-BoldInitializeFileContents', 'Procedure ( StringList : TStringList)
10932:   SIRegister_TBoldFileHandler(CL);
10933:   SIRegister_TBoldDiskFileHandler(CL);
10934:   Procedure BoldCloseAllFilehandlers
10935:   Procedure BoldRemoveUnchangedFilesFromEditor
10936:   Function BoldFileHandlerList : TBoldObjectArray
10937:   Function BoldFileHandlerForFile(path,FileName:String; ModuleType:TBoldModuleType;ShowInEditor:Bool;
OnInitializeFileContents : TBoldInitializeFileContents) : TBoldFileHandler

```

```

10938: end;
10939:
10940: procedure SIRegister_BoldWinINet(CL: TPSPascalCompiler);
10941: begin
10942:   PCharArr, 'array of PChar
10943:   Function BoldInternetOpen(Agent:String;
AccessType:integer;Proxy:String;ProxyByPass:String;Flags:integer):ptr);
10944:   Function BoldInternetOpenUrl(iNet:Pointer;URL: string; Headers:String;Flags,Context:cardinal):Pointer
10945:   Function BoldInternetReadFile(hFile:Pointer;Buff:Ptr;NumbofBytesToRead:Card;var
NumberOfBytesRead:Card):LongBool;
10946:   Function BoldInternetCloseHandle( HINet : Pointer ) : LongBool
10947:   Function BoldHttpRequestInfo( hRequest : Pointer; InfoLevel : Cardinal; Buffer : Pointer; BufferLength :
Cardinal; Reserved : Cardinal ) : LongBool
10948:   Function BoldInternetQueryDataAvailable( hFile : Pointer; var NumberOfBytesAvailable : Cardinal; flags :
Cardinal; Context : Cardinal ) : LongBool
10949:   Function BoldHttpRequest(hConnect: Pointer; Verb, ObjectName, Version, Referrer : String; AcceptTypes
: PCharArr; Flags, Context : Cardinal ) : Pointer
10950:   Function BoldHttpRequest(hRequest:Ptr;Headers:string;Optional:Ptr;Optionallength:Cardinal): LongBool
10951:   Function BoldInternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:Ptr):DWORD
10952:   Function BoldInternetAttemptConnect( dwReserved : DWORD ) : DWORD
10953:   Function BoldInternetConnect(hInet: HINTERNET;ServerName:string; nServerPort:INTERNET_PORT;
Username:string; Password : string; dwService : DWORD; dwFlags : DWORD; dwContext : DWORD):HINTERNET
10954:   Function BoldInternetCrackUrl(Url:PChar;UrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents):BOOL;
10955: end;
10956:
10957: procedure SIRegister_BoldQueryUserDlg(CL: TPSPascalCompiler);
10958: begin
10959:   TBoldQueryResult, '( qrYesAll, qrYes, qrNo, qrNoAll )
10960:   SIRegister_TfrmBoldQueryUser(CL);
10961:   Function QueryUser( const Title, Query : string ) : TBoldQueryResult
10962: end;
10963:
10964: (*-----*)
10965: procedure SIRegister_BoldQueue(CL: TPSPascalCompiler);
10966: begin
10967:   (('befIsInDisplayList','').SetString( BoldElementFlag0);
10968:   (('befStronglyDependedOfPrioritized','').SetString( BoldElementFlag1);
10969:   (('befFollowerSelected','').SetString( BoldElementFlag2);
10970:   FindClass('TOBJECT'),'TBoldQueue
10971:   FindClass('TOBJECT'),'TBoldQueueable
10972:   TBoldQueueDisplayMode', '( dmDisplayOne, dmDisplayAll )
10973:   SIRegister_TBoldQueueable(CL);
10974:   SIRegister_TBoldQueue(CL);
10975:   Function BoldQueueFinalized : Boolean
10976:   Function BoldInstalledQueue : TBoldQueue
10977: end;
10978:
10979: procedure SIRegister_Barcode(CL: TPSPascalCompiler);
10980: begin
10981:   const mmPerInch,'Extended').setExtended( 25.4);
10982:   TBarcodeType, '( bcCode_2_5_interleaved, bcCode_2_5_industrial, '
10983:   + ' bcCode_2_5_matrix, bcCode39, bcCode39Extended, bcCode128A, bcCode128B, bc'
10984:   + 'Code128C, bcCode93, bcCode93Extended, bcCodeMSI, bcCodePostNet, bcCodeCoda'
10985:   + 'bar, bcCodeEAN8, bcCodeEAN13, bcCodeUPC_A, bcCodeUPC_E0, bcCodeUPC_E1, bcC'
10986:   + 'odeUPC_Supp2, bcCodeUPC_Supp5, bcCodeEAN128A, bcCodeEAN128B, bcCodeEAN128C
10987:   TBarLineType, '( white, black, black_half )
10988:   TBarcodeOption, '( bcoNone, bcoCode, bcoTyp, bcoBoth )
10989:   TShowTextPosition, '( stpTopLeft, stpTopRight, stpTopCenter, st'
10990:   + 'pBottomLeft, stpBottomRight, stpBottomCenter )
10991:   TChecksumMethod, '( csmNone, csmModulo10 )
10992:   SIRegister_TAsBarcode(CL);
10993:   Function CheckSumModulo10( const data : string ) : string
10994:   Function ConvertMmToPixelsX( const Value : Double ) : Integer
10995:   Function ConvertMmToPixelsY( const Value : Double ) : Integer
10996:   Function ConvertInchToPixelsX( const Value : Double ) : Integer
10997:   Function ConvertInchToPixelsY( const Value : Double ) : Integer
10998: end;
10999:
11000: procedure SIRegister_Geometry(CL: TPSPascalCompiler); //OpenGL
11001: begin
11002:   THomogeneousByteVector, 'array[0..3] of Byte
11003:   THomogeneousWordVector, 'array[0..3] of Word
11004:   THomogeneousIntVector, 'array[0..3] of Integer
11005:   THomogeneousFltVector, 'array[0..3] of single
11006:   THomogeneousDblVector, 'array[0..3] of double
11007:   THomogeneousExtVector, 'array[0..3] of extended
11008:   TAffineByteVector, 'array[0..2] of Byte
11009:   TAffineWordVector, 'array[0..2] of Word
11010:   TAffineIntVector, 'array[0..2] of Integer
11011:   TAffineFltVector, 'array[0..2] of single
11012:   TAffineDblVector, 'array[0..2] of double
11013:   TAffineExtVector, 'array[0..2] of extended
11014:   THomogeneousByteMatrix, 'array[0..3] of THomogeneousByteVector
11015:   THomogeneousWordMatrix, 'array[0..3] of THomogeneousWordVector
11016:   THomogeneousIntMatrix, 'array[0..3] of THomogeneousIntVector
11017:   THomogeneousFltMatrix, 'array[0..3] of THomogeneousFltVector
11018:   THomogeneousDblMatrix, 'array[0..3] of THomogeneousDblVector
11019:   THomogeneousExtMatrix, 'array[0..3] of THomogeneousExtVector
11020:   TAffineByteMatrix, 'array[0..2] of TAffineByteVector

```

```

11021: TAffineWordMatrix', 'array[0..2] of TAffineWordVector
11022: TAffineIntMatrix', 'array[0..2] of TAffineIntVector
11023: TAffineFltMatrix', 'array[0..3] of TAffineFltVector
11024: TAffineDblMatrix', 'array[0..3] of TAffineDblVector
11025: TAffineExtMatrix', 'array[0..3] of TAffineExtVector
11026: TMatrix4b', THomogeneousByteMatrix
11027: TMatrix4w', THomogeneousWordMatrix
11028: TMatrix4i', THomogeneousIntMatrix
11029: TMatrix4f', THomogeneousFltMatrix
11030: TMatrix4d', THomogeneousDblMatrix
11031: TMatrix4e', THomogeneousExtMatrix
11032: TMatrix3b', TAffineByteMatrix
11033: TMatrix3w', TAffineWordMatrix
11034: TMatrix3i', TAffineIntMatrix
11035: TMatrix3f', TAffineFltMatrix
11036: TMatrix3d', TAffineDblMatrix
11037: TMatrix3e', TAffineExtMatrix
11038: //PMatrix', '^TMatrix // will not work
11039: TMatrixGL', THomogeneousFltMatrix
11040: THomogeneousMatrix', THomogeneousFltMatrix
11041: TAffineMatrix', TAffineFltMatrix
11042: TQuaternion', record Vector : TVector4f; end
11043: TRectangle', record Left : integer; Top : integer; Width : inte
11044: ger; Height : Integer; end
11045: TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
11046: + 'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
11047: + 'Z, ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )
11048: 'EPSILON', 'Extended').setExtended( 1E-100);
11049: 'EPSILON2', 'Extended').setExtended( 1E-50);
11050: Function VectorAddGL( V1, V2 : TVectorGL) : TVectorGL
11051: Function VectorAffineAdd( V1, V2 : TAffineVector) : TAffineVector
11052: Function VectorAffineCombine(V1,V2:TAffineVector; F1, F2 : Single) : TAffineVector
11053: Function VectorAffineDotProduct( V1, V2 : TAffineVector) : Single
11054: Function VectorAffineLerp( V1, V2 : TAffineVector; t : Single) : TAffineVector
11055: Function VectorAffineSubtract( V1, V2 : TAffineVector) : TAffineVector
11056: Function VectorAngle( V1, V2 : TAffineVector) : Single
11057: Function VectorCombine( V1, V2 : TVectorGL; F1, F2 : Single) : TVectorGL
11058: Function VectorCrossProduct( V1, V2 : TAffineVector) : TAffineVector
11059: Function VectorDotProduct( V1, V2 : TVectorGL) : Single
11060: Function VectorLength( V : array of Single) : Single
11061: Function VectorLerp( V1, V2 : TVectorGL; t : Single) : TVectorGL
11062: Procedure VectorNegate( V : array of Single)
11063: Function VectorNorm( V : array of Single) : Single
11064: Function VectorNormalize( V : array of Single) : Single
11065: Function VectorPerpendicular( V, N : TAffineVector) : TAffineVector
11066: Function VectorReflect( V, N : TAffineVector) : TAffineVector
11067: Procedure VectorRotate( var Vector : TVector4f; Axis : TVector3f; Angle : Single)
11068: Procedure VectorScale( V : array of Single; Factor : Single)
11069: Function VectorSubtractGL( V1, V2 : TVectorGL) : TVectorGL
11070: Function CreateRotationMatrixX( Sine, Cosine : Single) : TMatrixGL
11071: Function CreateRotationMatrixY( Sine, Cosine : Single) : TMatrixGL
11072: Function CreateRotationMatrixZ( Sine, Cosine : Single) : TMatrixGL
11073: Function CreateScaleMatrix( V : TAffineVector) : TMatrixGL
11074: Function CreateTranslationMatrix( V : TVectorGL) : TMatrixGL
11075: Procedure MatrixAdjoint( var M : TMatrixGL)
11076: Function MatrixAffineDeterminant( M : TAffineMatrix) : Single
11077: Procedure MatrixAffineTranspose( var M : TAffineMatrix)
11078: Function MatrixDeterminant( M : TMatrixGL) : Single
11079: Procedure MatrixInvert( var M : TMatrixGL)
11080: Function MatrixMultiply( M1, M2 : TMatrixGL) : TMatrixGL
11081: Procedure MatrixScale( var M : TMatrixGL; Factor : Single)
11082: Procedure MatrixTranspose( var M : TMatrixGL)
11083: Function QuaternionConjugate( Q : TQuaternion) : TQuaternion
11084: Function QuaternionFromPoints( V1, V2 : TAffineVector) : TQuaternion
11085: Function QuaternionMultiply( qL, qR : TQuaternion) : TQuaternion
11086: Function QuaternionSlerp( QStart,QEnd:TQuaternion; Spin:Integer; t:Single):TQuaternion
11087: Function QuaternionToMatrix( Q : TQuaternion) : TMatrixGL
11088: Procedure QuaternionToPoints( Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
11089: Function ConvertRotation( Angles : TAffineVector) : TVectorGL
11090: Function CreateRotationMatrix( Axis : TVector3f; Angle : Single) : TMatrixGL
11091: //Function MatrixDecompose( M : TMatrixGL; var Tran : TTransformations) : Boolean
11092: Function VectorAffineTransform( V : TAffineVector; M : TAffineMatrix) : TAffineVector
11093: Function VectorTransform( V : TVector4f; M : TMatrixGL) : TVector4f;
11094: Function VectorTransforml( V : TVector3f; M : TMatrixGL) : TVector3f;
11095: Function MakeAffineDblVector( V : array of Double) : TAffineDblVector
11096: Function MakeDblVector( V : array of Double) : THomogeneousDblVector
11097: Function MakeAffineVector( V : array of Single) : TAffineVector
11098: Function MakeQuaternion( Imag : array of Single; Real : Single) : TQuaternion
11099: Function MakeVector( V : array of Single) : TVectorGL
11100: Function PointInPolygonGL( xp, yp : array of Single; x, y : Single) : Boolean
11101: Function VectorAffineDblToFlt( V : TAffineDblVector) : TAffineVector
11102: Function VectorDblToFlt( V : THomogeneousDblVector) : THomogeneousVector
11103: Function VectorAffineFltToDbl( V : TAffineVector) : TAffineDblVector
11104: Function VectorFltToDbl( V : TVectorGL) : THomogeneousDblVector
11105: Function ArcCosGL( X : Extended) : Extended
11106: Function ArcSinGL( X : Extended) : Extended
11107: Function ArcTan2GL( Y, X : Extended) : Extended
11108: Function CoTanGL( X : Extended) : Extended
11109: Function DegToRadGL( Degrees : Extended) : Extended

```



```

11110: Function RadToDegGL( Radians : Extended) : Extended
11111: Procedure SinCosGL( Theta : Extended; var Sin, Cos : Extended)
11112: Function TanGL( X : Extended) : Extended
11113: Function Turn( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11114: Function Turn1( Matrix : TMatrixGL; MasterUp : TAffineVector; Angle: Single): TMatrixGL;
11115: Function Pitch( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11116: Function Pitch1( Matrix : TMatrixGL; MasterRight:TAffineVector;Angle:Single):TMatrixGL;
11117: Function Roll( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11118: Function Roll1( Matrix:TMatrixGL; MasterDirection:TAffineVector;Angle:Single):TMatrixGL;
11119: end;
11120:
11121:
11122: procedure SIRegister_JclRegistry(CL: TPSPascalCompiler);
11123: begin
11124:   Function RegCreateKey( const RootKey : HKEY; const Key, Value : string) : Longint
11125:   Function RegDeleteEntry( const RootKey : HKEY; const Key, Name : string) : Boolean
11126:   Function RegDeleteKeyTree( const RootKey : HKEY; const Key : string) : Boolean
11127:   Function RegReadBool( const RootKey : HKEY; const Key, Name : string) : Boolean
11128:   Function RegReadBoolDef( const RootKey : HKEY; const Key, Name : string; Def : Boolean) : Boolean
11129:   Function RegReadInteger( const RootKey : HKEY; const Key, Name : string) : Integer
11130:   Function RegReadIntegerDef( const RootKey : HKEY; const Key, Name : string; Def : Integer) : Integer
11131:   Function RegReadString( const RootKey : HKEY; const Key, Name : string) : string
11132:   Function RegReadStringDef( const RootKey : HKEY; const Key, Name, Def : string) : string
11133:   Function RegReadDWORD( const RootKey : HKEY; const Key, Name : string) : Int64
11134:   Function RegReadDWORDDef( const RootKey : HKEY; const Key, Name : string; Def : Int64) : Int64
11135:   Procedure RegWriteBool( const RootKey : HKEY; const Key, Name : string; Value : Boolean)
11136:   Procedure RegWriteInteger( const RootKey : HKEY; const Key, Name : string; Value : Integer)
11137:   Procedure RegWriteString( const RootKey : HKEY; const Key, Name, Value : string)
11138:   Procedure RegWriteDWORD( const RootKey : HKEY; const Key, Name : string; Value : Int64)
11139:   Function RegGetValueNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11140:   Function RegGetKeyNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11141:   Function RegHasSubKeys( const RootKey : HKEY; const Key : string) : Boolean
11142:   Function RegKeyExists( const RootKey : HKEY; const Key : string) : Boolean
11143:   AddTypeS('TExecKind', '( ekMachineRun, ekMachineRunOnce, ekUserRun, ekUser'
11144:     + 'RunOnce, ekServiceRun, ekServiceRunOnce )
11145:   AddClassN(FindClass('TOBJECT'), 'EJclRegistryError
11146:   Function UnregisterAutoExec( ExecKind : TExecKind; const Name : string) : Boolean
11147:   Function RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline : string) : Boolean
11148:   Function RegSaveList(const RootKey:HKEY;const Key:string; const ListName:string;const
Items:TStrings):Bool;
11149:   Function RegLoadList(const RootKey:HKEY;const Key:string;const ListName:string;const
SaveTo:TStrings):Bool;
11150:   Function RegDelList( const RootKey:HKEY;const Key:string; const ListName:string) : Boolean
11151: end;
11152:
11153: procedure SIRegister_JclCOM(CL: TPSPascalCompiler);
11154: begin
11155:   CLSID_StdComponentCategoriesMgr, 'TGUID').SetString( '{0002E005-0000-0000-C000-000000000046}'
11156:   CATID_SafeForInitializing, 'TGUID').SetString( '{7DD95802-9882-11CF-9FA9-00AA006C42C4}'
11157:   CATID_SafeForScripting, 'TGUID').SetString( '{7DD95801-9882-11CF-9FA9-00AA006C42C4}'
11158:   icMAX_CATEGORY_DESC_LEN, 'LongInt').SetInt( 128);
11159:   FindClass('TOBJECT'), 'EInvalidParam
11160:   Function IsDCOMInstalled : Boolean
11161:   Function IsDCOMEnabled : Boolean
11162:   Function GetDCOMVersion : string
11163:   Function GetMDACVersion : string
11164:   Function MarshalInterThreadInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
VarArray:OleVariant):HResult;
11165:   Function MarshalInterProcessInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11166:   Function MarshalInterProcessInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
VarArray:OleVariant):HResult;
11167:   Function MarshalInterMachineInterfaceInStream( const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11168:   Function MarshalInterMachineInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
VarArray:OleVariant):HResult;
11169:   Function CreateComponentCategory( const CatID : TGUID; const sDescription : string) : HResult
11170:   Function RegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11171:   Function UnRegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11172:   Function ResetIStreamToStart( Stream : IStream) : Boolean
11173:   Function SizeOfIStreamContents( Stream : IStream) : Largeint
11174:   Function StreamToVariantArray( Stream : TStream) : OleVariant;
11175:   Function StreamToVariantArray1( Stream : IStream) : OleVariant;
11176:   Procedure VariantArrayToStream( VarArray : OleVariant; var Stream : TStream);
11177:   Procedure VariantArrayToStream1( VarArray : OleVariant; var Stream : IStream);
11178: end;
11179:
11180:
11181: procedure SIRegister_JclUnitConv_mx2(CL: TPSPascalCompiler);
11182: begin
11183:   Const ('CelsiusFreezingPoint', 'Extended').setExtended( 0.0);
11184:   FahrenheitFreezingPoint, 'Extended').setExtended( 32.0);
11185:   KelvinFreezingPoint, 'Extended').setExtended( 273.15);
11186:   CelsiusAbsoluteZero, 'Extended').setExtended( - 273.15);
11187:   FahrenheitAbsoluteZero, 'Extended').setExtended( - 459.67);
11188:   KelvinAbsoluteZero, 'Extended').setExtended( 0.0);
11189:   DegPerCycle, 'Extended').setExtended( 360.0);
11190:   DegPerGrad, 'Extended').setExtended( 0.9);
11191:   DegPerRad, 'Extended').setExtended( 57.295779513082320876798154814105);
11192:   GradPerCycle, 'Extended').setExtended( 400.0);
11193:   GradPerDeg, 'Extended').setExtended( 1.111111111111111111111111111111);

```

```

11194: GradPerRad('Extended').setExtended( 63.661977236758134307553505349006);
11195: RadPerCycle('Extended').setExtended( 6.283185307179586476925286766559);
11196: RadPerDeg('Extended').setExtended( 0.017453292519943295769236907684886);
11197: RadPerGrad('Extended').setExtended( 0.015707963267948966192313216916398);
11198: CyclePerDeg('Extended').setExtended( 0.0027777777777777777777777777777);
11199: CyclePerGrad('Extended').setExtended( 0.0025);
11200: CyclePerRad('Extended').setExtended( 0.15915494309189533576888376337251);
11201: ArcMinutesPerDeg('Extended').setExtended( 60.0);
11202: ArcSecondsPerArcMinute('Extended').setExtended( 60.0);
11203: Function HowAOneLinerCanBiteYou( const Step, Max : Longint ) : Longint
11204: Function MakePercentage( const Step, Max : Longint ) : Longint
11205: Function CelsiusToKelvin( const T : double ) : double
11206: Function CelsiusToFahrenheit( const T : double ) : double
11207: Function KelvinToCelsius( const T : double ) : double
11208: Function KelvinToFahrenheit( const T : double ) : double
11209: Function FahrenheitToCelsius( const T : double ) : double
11210: Function FahrenheitToKelvin( const T : double ) : double
11211: Function CycleToDeg( const Cycles : double ) : double
11212: Function CycleToGrad( const Cycles : double ) : double
11213: Function CycleToRad( const Cycles : double ) : double
11214: Function DegToCycle( const Degrees : double ) : double
11215: Function DegToGrad( const Degrees : double ) : double
11216: Function DegToRad( const Degrees : double ) : double
11217: Function GradToCycle( const Grads : double ) : double
11218: Function GradToDeg( const Grads : double ) : double
11219: Function GradToRad( const Grads : double ) : double
11220: Function RadToCycle( const Radians : double ) : double
11221: Function RadToDeg( const Radians : double ) : double
11222: Function RadToGrad( const Radians : double ) : double
11223: Function DmsToDeg( const D, M : Integer; const S : double ) : double
11224: Function DmsToRad( const D, M : Integer; const S : double ) : double
11225: Procedure DegToDms( const Degrees : double; out D, M : Integer; out S : double)
11226: Function DegToDmsStr( const Degrees : double; const SecondPrecision : Cardinal ) : string
11227: Procedure CartesianToPolar( const X, Y : double; out R, Phi : double)
11228: Procedure PolarToCartesian( const R, Phi : double; out X, Y : double)
11229: Procedure CartesianToCylinder( const X, Y, Z : double; out R, Phi, Zeta : double)
11230: Procedure CartesianToSpheric( const X, Y, Z : double; out Rho, Phi, Theta : double)
11231: Procedure CylinderToCartesian( const R, Phi, Zeta : double; out X, Y, Z : double)
11232: Procedure SphericToCartesian( const Rho, Phi, Theta : double; out X, Y, Z : double)
11233: Function CmToInch( const Cm : double ) : double
11234: Function InchToCm( const Inch : double ) : double
11235: Function FeetToMetre( const Feet : double ) : double
11236: Function MetreToFeet( const Metre : double ) : double
11237: Function YardToMetre( const Yard : double ) : double
11238: Function MetreToYard( const Metre : double ) : double
11239: Function NmToKm( const Nm : double ) : double
11240: Function KmToNm( const Km : double ) : double
11241: Function KmToSm( const Km : double ) : double
11242: Function SmToKm( const Sm : double ) : double
11243: Function LitreToGalUs( const Litre : double ) : double
11244: Function GalUsToLitre( const GalUs : double ) : double
11245: Function GalUsToGalCan( const GalUs : double ) : double
11246: Function GalCanToGalUs( const GalCan : double ) : double
11247: Function GalUsToGalUk( const GalUs : double ) : double
11248: Function GalUkToGalUs( const GalUk : double ) : double
11249: Function LitreToGalCan( const Litre : double ) : double
11250: Function GalCanToLitre( const GalCan : double ) : double
11251: Function LitreToGalUk( const Litre : double ) : double
11252: Function GalUkToLitre( const GalUk : double ) : double
11253: Function KgToLb( const Kg : double ) : double
11254: Function LbToKg( const Lb : double ) : double
11255: Function KgToOz( const Kg : double ) : double
11256: Function OzToKg( const Oz : double ) : double
11257: Function CwtUsToKg( const Cwt : double ) : double
11258: Function CwtUkToKg( const Cwt : double ) : double
11259: Function KaratToKg( const Karat : double ) : double
11260: Function KgToCwtUs( const Kg : double ) : double
11261: Function KgToCwtUk( const Kg : double ) : double
11262: Function KgToKarat( const Kg : double ) : double
11263: Function KgToSton( const Kg : double ) : double
11264: Function KgToLton( const Kg : double ) : double
11265: Function StonToKg( const STon : double ) : double
11266: Function LtonToKg( const Lton : double ) : double
11267: Function QrUsToKg( const Qr : double ) : double
11268: Function QrUkToKg( const Qr : double ) : double
11269: Function KgToQrUs( const Kg : double ) : double
11270: Function KgToQrUk( const Kg : double ) : double
11271: Function PascalToBar( const Pa : double ) : double
11272: Function PascalToAt( const Pa : double ) : double
11273: Function PascalToTorr( const Pa : double ) : double
11274: Function BarToPascal( const Bar : double ) : double
11275: Function AtToPascal( const At : double ) : double
11276: Function TorrToPascal( const Torr : double ) : double
11277: Function KnotToMs( const Knot : double ) : double
11278: Function HpElectricToWatt( const HpE : double ) : double
11279: Function HpMetricToWatt( const HpM : double ) : double
11280: Function MsToKnot( const ms : double ) : double
11281: Function WattToHpElectric( const W : double ) : double
11282: Function WattToHpMetric( const W : double ) : double

```

```

11283: function getBigPI: string; //PI of 1000 numbers
11284:
11285: procedure SIRegister_devcutils(CL: TPSPascalCompiler);
11286: begin
11287:   Function CDEExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer ) : THandle
11288:   Procedure CDCopyFile( const FileName, DestName : string)
11289:   Procedure CDMoveFile( const FileName, DestName : string)
11290:   Function MakeCommaTextToColor( Text : string; Index : Integer; DefaultColor : TColor ) : TColor
11291:   Procedure CDDeleteFiles( Sender : TObject; s : string)
11292:   Function CDGetTempDir : string
11293:   Function CDGetFileSize( FileName : string ) : longint
11294:   Function GetFileTime( FileName : string ) : longint
11295:   Function GetShortName( FileName : string ) : string
11296:   Function GetFullName( FileName : string ) : string
11297:   Function WinReboot : boolean
11298:   Function WinDir : String
11299:   Function RunFile( FileToRun : string; Params : string; Dir : string; Wait : boolean ) : cardinal
11300:   Function RunFile_( Cmd, WorkDir : string; Wait : boolean ) : Boolean
11301:   Function devExecutor : TdevExecutor
11302: end;
11303:
11304: procedure SIRegister_FileAssocs(CL: TPSPascalCompiler);
11305: begin
11306:   Procedure CheckAssociations // AssociationsCount', 'LongInt').SetInt( 7);
11307:   Procedure Associate( Index : integer)
11308:   Procedure UnAssociate( Index : integer)
11309:   Function IsAssociated( Index : integer ) : boolean
11310:   Function CheckFileType( const extension, filetype, description, verb, serverapp : string ) : boolean
11311:   Procedure RegisterFileType( const extension, filetype, description, verb, serverapp, IcoNum: string)
11312:   Procedure RegisterDDEServer( const filetype, verb, topic, servername, macro : string)
11313:   procedure RefreshIcons;
11314:   function GetShadeColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11315:   function MergColor(Colors: Array of TColor): TColor;
11316:   function NewColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11317:   procedure DimBitmap(ABitmap: TBitmap; Value: integer);
11318:   function GrayColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11319:   function GetInverseColor(AColor: TColor): TColor;
11320:   procedure GrayBitmap(ABitmap: TBitmap; Value: integer);
11321:   procedure DrawBitmapShadow(B: TBitmap; ACanvas: TCanvas; X,Y: integer; ShadowColor: TColor);
11322:   procedure DrawCheckMark(ACanvas: TCanvas; X, Y: integer);
11323:   Procedure GetSystemMenuFont(Font: TFont);
11324: end;
11325:
11326: //*****unit uPSI_JvHLParse;*****
11327: function IsStringConstant(const St: string): Boolean;
11328: function IsIntConstant(const St: string): Boolean;
11329: function IsRealConstant(const St: string): Boolean;
11330: function IsIdentifier(const ID: string): Boolean;
11331: function GetStringValue(const St: string): string;
11332: procedure ParseString(const S: string; Ss: TStrings);
11333: function IsStringConstantW(const St: WideString): Boolean;
11334: function IsIntConstantW(const St: WideString): Boolean;
11335: function IsRealConstantW(const St: WideString): Boolean;
11336: function IsIdentifierW(const ID: WideString): Boolean;
11337: function GetStringValueW(const St: WideString): WideString;
11338: procedure ParseStringW(const S: WideString; Ss: TStrings);
11339:
11340:
11341: //*****unit uPSI_JclMapi;*****
11342:
11343: Function JclSimpleSendMail( const ARecipient, AName, ASubject, ABody : string; const AAttachment :
  TFileName; ShowDialog : Boolean; AParentWND : HWND ) : Boolean
11344: Function JclSimpleSendFax( const ARecipient, AName, ASubject, ABody : string; const AAttachment :
  TFileName; ShowDialog : Boolean; AParentWND : HWND ) : Boolean
11345: Function JclSimpleBringUpSendMailDialog(const ASubject, ABody: string; const
  AAttach: TFileName; AParentWND: HWND): Bool
11346: Function MapiCheck( const Res : DWORD; IgnoreUserAbort : Boolean ) : DWORD
11347: Function MapiErrorMessage( const ErrorCode : DWORD ) : string
11348:
11349: procedure SIRegister_IdNTLM(CL: TPSPascalCompiler);
11350: begin
11351:   // 'Pdes_key_schedule', '^des_key_schedule // will not work
11352:   Function BuildType1Message( ADomain, AHost : String ) : String
11353:   Function BuildType3Message(ADomain, AHost, AUsername: WideString; APassword, ANonce: String): String
11354:   Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
11355:   Function FindAuthClass( AuthName : String ) : TIdAuthenticationClass
11356:   GBase64CodeTable(, 'string').SetString(' ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
11357:   GXXECodeTable(, 'string').SetString('+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
11358:   GUUECodeTable(, 'string').SetString('!'#$%&'()*+,-./0123456789;=<=?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
11359: end;
11360:
11361: procedure SIRegister_WDosSocketUtils(CL: TPSPascalCompiler);
11362: begin
11363:   ('IpAny', 'LongWord').SetUInt( $00000000);
11364:   IpLoopBack(, 'LongWord').SetUInt( $7F000001);
11365:   IpBroadcast(, 'LongWord').SetUInt( $FFFFFFF);
11366:   IpNone(, 'LongWord').SetUInt( $FFFFFFF);
11367:   PortAny(, 'LongWord').SetUInt( $0000);
11368:   SocketMaxConnections(, 'LongInt').SetInt( 5);

```

```

11369: TIpAddr', 'LongWord
11370: TIpRec', 'record IpB1 : byte; IpB2 : byte; IpB3 : byte; IpB4 : Byte; end
11371: Function HostToNetLong( HostLong : LongWord ) : LongWord
11372: Function HostToNetShort( HostShort : Word ) : Word
11373: Function NetToHostLong( NetLong : LongWord ) : LongWord
11374: Function NetToHostShort( NetShort : Word ) : Word
11375: Function StrToIp( Ip : string ) : TIpAddr
11376: Function IpToStr( Ip : TIpAddr ) : string
11377: end;
11378:
11379: (*-----*)
11380: procedure SIRegister_ALSMTPClient(CL: TPSPascalCompiler);
11381: begin
11382:   TALSMTPClientAuthType', '( AlsmtpClientAuthNone, alsmtpClientAut
11383:     + 'hPlain, AlsmtpClientAuthLogin, AlsmtpClientAuthCramMD5, AlsmtpClientAuthCr
11384:     + 'amShal, AlsmtpClientAuthAutoSelect )
11385:   TALSMTPClientAuthTypeSet', 'set of TALSMTPClientAuthType
11386:   SIRegister_TALSMTPClient(CL);
11387: end;
11388:
11389: procedure SIRegister_WDosPlcUtils(CL: TPSPascalCompiler);
11390: begin
11391:   TBitNo', 'Integer
11392:   TStByteNo', 'Integer
11393:   TStationNo', 'Integer
11394:   TInOutNo', 'Integer
11395:   TIO', '( EE, AA, NE, NA )
11396:   TBitSet', 'set of TBitNo
11397:   TAddrKind', 'set of ( akBit0, akBit1, akBit2, akOut, akNot, akBus )
11398:   TBitAddrRec', 'record Kind : TAddrKind; InOutNo : TInOutNo; ByteNo : Byte; end
11399:   TBitAddr', 'LongInt
11400:   TByteAddrRec', 'record Kind : TAddrKind; ByteNo : Byte; end
11401:   TByteAddr', 'SmallInt
11402:   TInOutState', '( iosInit, iosHalt, iosRun, iosError )
11403:   Function BitAddr(aIo: TIO; aInOutNo : TInOutNo; aByteNo : Byte; aBitNo : TBitNo) : TBitAddr
11404:   Function BusBitAddr(aIo:TIO;aInOutNo:TInOutNo;aStat:TStatNo;aStByteNo:TStByteNo;aBitNo:TBitNo):TBitAddr;
11405:   Procedure BitAddrToValues(aBitAdr:TBitAdr;var aIo:TIO;var aInOutNo:TInOutNo;var aByteNo:Byte;var
aBitNo:TBitNo);
11406:   Function BitAddrToStr( Value : TBitAddr ) : string
11407:   Function StrToBitAddr( const Value : string ) : TBitAddr
11408:   Function ByteAddr( aIo : TIO; aInOutNo : TInOutNo; aByteNo : Byte ) : TByteAddr
11409:   Function BusByteAddr(aIo:TIO;aInOutNo:TInOutNo;aStation:TStationNo;aStByteNo: TStByteNo):TByteAddr
11410:   Procedure ByteAddrToValues(aByteAddr:TByteAddr;var aIo:TIO;var aInOutNo:TInOutNo;var aByteNo:Byte)
11411:   Function ByteAddrToStr( Value : TByteAddr ) : string
11412:   Function StrToByteAddr( const Value : string ) : TByteAddr
11413:   Procedure IncByteAddr( var ByteAddr : TByteAddr; Increment : Integer)
11414:   Procedure DecByteAddr( var ByteAddr : TByteAddr; Decrement : Integer)
11415:   Function InOutStateToStr( State : TInOutState ) : string
11416:   Function MasterErrorToStr( ErrorCode : TErrorCode ) : string
11417:   Function SlaveErrorToStr( ErrorCode : TErrorCode ) : string
11418: end;
11419:
11420: procedure SIRegister_WDosTimers(CL: TPSPascalCompiler);
11421: begin
11422:   TIntFreq', '( ifNone, if32768, if16384, if8192, if4096, if2048, '
11423:     + 'if1024, if512, if256, if128, if64, if32, if16, if8, if4, if2 )
11424:   DpmiPmVector', 'Int64
11425:   'DInterval', 'LongInt').SetInt( 1000);
11426:   //'DEnabled', 'Boolean')BoolToStr( True);
11427:   'DIntFreq', 'string').SetString( if64);
11428:   //'DMessages', 'Boolean').SetString( if64);
11429:   SIRegister_TwdxCuomTimer(CL);
11430:   SIRegister_TwdXTimer(CL);
11431:   SIRegister_TwdXRtcTimer(CL);
11432:   SIRegister_TCustomIntTimer(CL);
11433:   SIRegister_TIntTimer(CL);
11434:   SIRegister_TRtcIntTimer(CL);
11435:   Function RealNow : TDateTime
11436:   Function MsToDateTime( MilliSecond : LongInt ) : TDateTime
11437:   Function DateTimeToMs( Time : TDateTime ) : LongInt
11438: end;
11439:
11440: procedure SIRegister_IdSysLogMessage(CL: TPSPascalCompiler);
11441: begin
11442:   TIdSyslogPRI', 'Integer
11443:   TIdSyslogFacility', '( sfKernel, sfUserLevel, sfMailSystem, sfSy
11444:     + 'stemDaemon, sfSecurityOne, sfSysLogInternal, sfLPR, sfNNTP, sfUUCP, sfCloc
11445:     + 'kDaemonOne, sfSecurityTwo, sfFTPDaemon, sfNTP, sfLogAudit, sfLogAlert, sfC
11446:     + 'lockDaemonTwo, sfLocalUseZero, sfLocalUseOne, sfLocalUseTwo, sfLocalUseThr
11447:     + 'ee, sfLocalUseFour, sfLocalUseFive, sfLocalUseSix, sfLocalUseSeven )
11448:   TIdSyslogSeverity', '(slEmergency,slAlert,slCritical,slError,slWarning,slNotice,slInformational,slDebug)
11449:   SIRegister_TIdSysLogMsgPart(CL);
11450:   SIRegister_TIdSysLogMessage(CL);
11451:   Function FacilityToString( AFac : TIdSyslogFacility ) : string
11452:   Function SeverityToString( ASec : TIdSyslogSeverity ) : string
11453:   Function NoToSeverity( ASev : Word ) : TIdSyslogSeverity
11454:   Function logSeverityToNo( ASev : TIdSyslogSeverity ) : Word
11455:   Function NoToFacility( AFac : Word ) : TIdSyslogFacility
11456:   Function logFacilityToNo( AFac : TIdSyslogFacility ) : Word

```



```

11457: end;
11458:
11459: procedure SIRegister_TextUtils(CL: TPSPascalCompiler);
11460: begin
11461:   'UWhitespace','String').SetString( '(?:\s*)
11462:   Function StripSpaces( const AText : string) : string
11463:   Function CharCount( const AText : string; Ch : Char) : Integer
11464:   Function BalancedText( const AText : string; const Ch1, Ch2 : Char; const Count : Integer) : string
11465:   Function BalancedTextReg( const AText:string; const Ch1, Ch2 : Char; const Count : Integer) : string
11466: end;
11467:
11468:
11469: procedure SIRegister_ExtPascalUtils(CL: TPSPascalCompiler);
11470: begin
11471:   ExtPascalVersion','String').SetString( '0.9.8
11472:   AddTypeS('TBrowser', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
11473:   + 'Opera, brKonqueror, brMobileSafari )
11474:   AddTypeS('TCSSUnit', '( cssPX, cssPerc, cssEM, cssEX, cssIN, cssCM, cssMM, cssPT, cssPC, cssnone )
11475:   AddTypeS('TExtProcedure', 'Procedure
11476:   Function DetermineBrowser( const UserAgentStr : string) : TBrowser
11477:   Function ExtExtract(const Delims:array of string;var S:string;var Matches:TStringList;Remove:bool):bool;
11478:   Function ExtExplode( Delim : char; const S : string; Separator : char) : TStringList
11479:   Function FirstDelimiter( const Delimiters, S : string; Offset : integer) : integer
11480:   Function RPosEx( const Substr, Str : string; Offset : integer) : integer
11481:   Function CountStr( const Substr, Str : string; UntilStr : string) : integer
11482:   Function StrToJS( const S : string; UseBR : boolean) : string
11483:   Function CaseOf( const S : string; const Cases : array of string) : integer
11484:   Function RCaseOf( const S : string; const Cases : array of string) : integer
11485:   Function EnumToJSSString( TypeInfo : PTypeInfo; Value : integer) : string
11486:   Function SetPaddings(Top:integer;Right:int;Bottom:intr;Left:integer;CSSUnit:TCSSUnit;Header:bool):string;
11487:   Function SetMargins(Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool): string;
11488:   Function ExtBefore( const BeforeS, AfterS, S : string) : boolean
11489:   Function IsUpperCase( S : string) : boolean
11490:   Function BeautifyJS(const AScript:string;const StartingLevel:integer;SplitHTMLNewLine: boolean):string;
11491:   Function BeautifyCSS( const AStyle : string) : string
11492:   Function LengthRegExp( Rex : string; CountAll : Boolean) : integer
11493:   Function JSDateToDateTime( JSDate : string) : TDateTime
11494: end;
11495:
11496: procedure SIRegister_JclShell(CL: TPSPascalCompiler);
11497: begin
11498:   TSHDeleteOption', '( doSilent, doAllowUndo, doFilesOnly )
11499:   TSHDeleteOptions', 'set of TSHDeleteOption
11500:   TSHRenameOption', '( roSilent, roRenameOnCollision )
11501:   TSHRenameOptions', 'set of TSHRenameOption
11502:   Function SHDeleteFiles( Parent : HWND; const Files : string; Options : TSHDeleteOptions) : Boolean
11503:   Function SHDeleteFolder( Parent : HWND; const Folder : string; Options : TSHDeleteOptions) : Boolean
11504:   Function SHRenameFile( const Src, Dest : string; Options : TSHRenameOptions) : Boolean
11505:   TEnumFolderFlag', '( efFolders, efNonFolders, efIncludeHidden )
11506:   TEnumFolderFlags', 'set of TEnumFolderFlag
11507:   TEnumFolderRec', 'record DisplayName : string; Attributes : DWORD'
11508:   + 'D; IconLarge : HICON; IconSmall : HICON; Item : PItemIdList; EnumIdList : '
11509:   + 'IEnumIdList; Folder : IShellFolder; end
11510:   Function SHEnumFolderFirst(const Folder:string;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Boolean;
11511:   Function SHEnumSpecialFolderFirst(SpecialFolder:DWORD;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11512:   Procedure SHEnumFolderClose( var F : TEnumFolderRec)
11513:   Function SHEnumFolderNext( var F : TEnumFolderRec) : Boolean
11514:   Function GetSpecialFolderLocation( const Folder : Integer) : string
11515:   Function DisplayPropDialog( const Handle : HWND; const FileName : string) : Boolean;
11516:   Function DisplayPropDialog1( const Handle : HWND; const Item : PItemIdList) : Boolean;
11517:   Function DisplayContextMenu( const Handle : HWND; const FileName : string; Pos : TPoint) : Boolean
11518:   Function OpenFolder( const Path : string; Parent : HWND) : Boolean
11519:   Function OpenSpecialFolder( FolderID : Integer; Parent : HWND) : Boolean
11520:   Function SHReallocMem( var P : Pointer; Count : Integer) : Boolean
11521:   Function SHAllocMem( out P : Pointer; Count : Integer) : Boolean
11522:   Function SHGetMem( var P : Pointer; Count : Integer) : Boolean
11523:   Function SHFreeMem( var P : Pointer) : Boolean
11524:   Function DriveToPidlBind( const DriveName : string; out Folder : IShellFolder) : PItemIdList
11525:   Function PathToPidl( const Path : string; Folder : IShellFolder) : PItemIdList
11526:   Function PathToPidlBind( const FileName : string; out Folder : IShellFolder) : PItemIdList
11527:   Function PidlBindToParent(const IdList:PItemIdList;out Folder:IShellFolder;out Last:PItemIdList):Bool;
11528:   Function PidlCompare( const Pidl1, Pidl2 : PItemIdList) : Boolean
11529:   Function PidlCopy( const Source : PItemIdList; out Dest : PItemIdList) : Boolean
11530:   Function PidlFree( var IdList : PItemIdList) : Boolean
11531:   Function PidlGetDepth( const Pidl : PItemIdList) : Integer
11532:   Function PidlGetLength( const Pidl : PItemIdList) : Integer
11533:   Function PidlGetNext( const Pidl : PItemIdList) : PItemIdList
11534:   Function PidlToPath( IdList : PItemIdList) : string
11535:   Function StrRetFreeMem( StrRet : TStrRet) : Boolean
11536:   Function StrRetToString( IdList : PItemIdList; StrRet : TStrRet; Free : Boolean) : string
11537:   PShellLink', '^TShellLink // will not work
11538:   TShellLink', 'record Arguments : string; ShowCmd : Integer; Work'
11539:   + 'ingDirectory : string; IdList : PItemIdList; Target : string; Description '
11540:   + ': string; IconLocation : string; IconIndex : Integer; HotKey : Word; end
11541:   Procedure ShellLinkFree( var Link : TShellLink)
11542:   Function ShellLinkResolve( const FileName : string; var Link : TShellLink) : HRESULT
11543:   Function ShellLinkCreate( const Link : TShellLink; const FileName : string) : HRESULT
11544:   Function ShellLinkCreateSystem(const Link:TShellLink;const Folder:Integer; const FileName:string):HRESULT;
11545:   Function ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon) : Boolean

```

```

11546: Function SHDllGetVersion( const FileName : string; var Version : TDllVersionInfo) : Boolean
11547: Function GetSystemIcon( IconIndex : Integer; Flags : Cardinal) : HICON
11548: Function OverlayIcon( var Icon : HICON; Overlay : HICON; Large : Boolean) : Boolean
11549: Function OverlayIconShortcut( var Large, Small : HICON) : Boolean
11550: Function OverlayIconShared( var Large, Small : HICON) : Boolean
11551: Function SHGetItemInfoTip( const Folder : IShellFolder; Item : PItemIdList) : string
11552: Function ShellExecEx(const FileName:string;const Parameters:string;const Verb:string; CmdShow:Int):Bool;
11553: Function ShellExec(Wnd: Integer;const Operation,FileName,Parameters,Directy:string;ShowCommand:Int):Bool;
11554: Function ShellExecAndWait(const FileName:string;const Paramets:string;const Verb:string;CmdShow:Int):Bool;
11555: Function ShellOpenAs( const FileName : string) : Boolean
11556: Function ShellRasDial( const EntryName : string) : Boolean
11557: Function ShellRunControlPanel( const NameOrFileName:string; AppletNumber:Integer):Boolean
11558: Function GetFileNameIcon( const FileName : string; Flags : Cardinal) : HICON
11559: TJclFileExeType, '( etError, etMsDos, etWin16, etWin32Gui, etWin32Con )
11560: Function GetFileExeType( const FileName : TFileName) : TJclFileExeType
11561: Function ShellFindExecutable( const FileName, DefaultDir : string) : string
11562: Procedure keybd_event( bVk : Byte; bScan : Byte; dwFlags, dwExtraInfo : DWORD)
11563: Function OemKeyScan( wOemChar : Word) : DWORD
11564: Procedure mouse_event( dwFlags, dx, dy, dwData, dwExtraInfo : DWORD)
11565: end;
11566:
11567: procedure SIRegister_cXMLFunctions(CL: TPSPascalCompiler);
11568: begin
11569:   xmlVersion', 'String').SetString( '1.0 FindClass('TOBJECT'),'Exml
11570:   //Function xmlValidChar( const Ch : AnsiChar) : Boolean;
11571: Function xmlValidChar1( const Ch : UCS4Char) : Boolean;
11572: Function xmlValidChar2( const Ch : WideChar) : Boolean;
11573: Function xmlIsSpaceChar( const Ch : WideChar) : Boolean
11574: Function xmlIsLetter( const Ch : WideChar) : Boolean
11575: Function xmlIsDigit( const Ch : WideChar) : Boolean
11576: Function xmlIsNameStartChar( const Ch : WideChar) : Boolean
11577: Function xmlIsNameChar( const Ch : WideChar) : Boolean
11578: Function xmlIsPubidChar( const Ch : WideChar) : Boolean
11579: Function xmlValidName( const Text : UnicodeString) : Boolean
11580: //xmlSpace', 'Char').SetString( # $20 or # $9 or # $D or # $A);
11581: //Function xmlSkipSpace( var P : PWideChar) : Boolean
11582: //Function xmlSkipEq( var P : PWideChar) : Boolean
11583: //Function xmlExtractQuotedText( var P : PWideChar; var S : UnicodeString) : Boolean
11584: //Function xmlGetEntityEncoding( const Buf : Pointer; const BufSize : Integer; out HeaderSize : Integer)
: TUnicodeCodecClass
11585: Function xmlResolveEntityReference( const RefName : UnicodeString) : WideChar
11586: Function xmlTag( const Tag : UnicodeString) : UnicodeString
11587: Function xmlEndTag( const Tag : UnicodeString) : UnicodeString
11588: Function xmlAttrTag( const Tag : UnicodeString; const Attr : UnicodeString) : UnicodeString
11589: Function xmlEmptyTag( const Tag, Attr : UnicodeString) : UnicodeString
11590: Procedure xmlSafeTextInPlace( var Txt : UnicodeString)
11591: Function xmlSafeText( const Txt : UnicodeString) : UnicodeString
11592: Function xmlSpaceIndent( const IndentLength : Integer; const IndentLevel : Integer):UnicodeString
11593: Function xmlTabIndent( const IndentLevel : Integer) : UnicodeString
11594: Function xmlComment( const Comment : UnicodeString) : UnicodeString
11595: Procedure SelfTestcXMLFunctions
11596: end;
11597:
11598: (*-----*)
11599: procedure SIRegister_DepWalkUtils(CL: TPSPascalCompiler);
11600: begin
11601:   Function AwaitCursor : IUnknown
11602:   Function ChangeCursor( NewCursor : TCursor) : IUnknown
11603:   Procedure SuspendRedraw( AControl : TWinControl; Suspend : boolean)
11604:   Function YesNo( const ACaption, AMsg : string) : boolean
11605:   Procedure strTokenize( const S : string; Delims : TSysCharSet; Results : TStrings)
11606:   Function GetBorlandLibPath( Version : integer; ForDelphi : boolean) : string
11607:   Function GetExpandedLibRoot( Version : integer; ForDelphi : boolean) : string
11608:   Procedure GetPathList( Version : integer; ForDelphi : boolean; Strings : TStrings)
11609:   Procedure GetSystemPaths( Strings : TStrings)
11610:   Procedure MakeEditNumeric( EditHandle : integer)
11611: end;
11612:
11613: procedure SIRegister_yuvconverts(CL: TPSPascalCompiler);
11614: begin
11615:   AddTypes( 'TVideoCodec', '(vcUnknown,vcRGB,vcYUY2,vcUYVY,vcBTYUV,vcYV,U9,vcYUV12,vcY8,vcY211)
11616:   'BI_YUY2', 'LongWord').SetUInt( $32595559);
11617:   'BI_UYVY', 'LongWord').SetUInt( $59565955);
11618:   'BI_BTUV', 'LongWord').SetUInt( $50313459);
11619:   'BI_YVU9', 'LongWord').SetUInt( $39555659);
11620:   'BI_YUV12', 'LongWord').SetUInt( $30323449);
11621:   'BI_Y8', 'LongWord').SetUInt( $20203859);
11622:   'BI_Y211', 'LongWord').SetUInt( $31313259);
11623:   Function BICompressionToVideoCodec( Value : DWord) : TVideoCodec
11624:   Function ConvertCodecToRGB(Coec:TVideoCodec;Src,Dst:Pointer;AWidth,AHeight:Integer):Boolean;
11625: end;
11626:
11627: (*-----*)
11628: procedure SIRegister_AviCap(CL: TPSPascalCompiler);
11629: begin
11630:   'WM_USER', 'LongWord').SetUInt( $0400);
11631:   'WM_CAP_START', 'LongWord').SetUInt($0400);
11632:   'WM_CAP_END', 'longword').SetUInt($0400+85);
11633:   //WM_CAP_START+ 85

```

```

11634: // WM_CAP_SET_CALLBACK_CAPCONTROL = (WM_CAP_START+ 85);
11635: Function capSetCallbackOnError( hwnd : THandle; fpProc : LongInt) : LongInt
11636: Function capSetCallbackOnStatus( hwnd : THandle; fpProc : LongInt) : LongInt
11637: Function capSetCallbackOnYield( hwnd : THandle; fpProc : LongInt) : LongInt
11638: Function capSetCallbackOnFrame( hwnd : THandle; fpProc : LongInt) : LongInt
11639: Function capSetCallbackOnVideoStream( hwnd : THandle; fpProc : LongInt) : LongInt
11640: Function capSetCallbackOnWaveStream( hwnd : THandle; fpProc : LongInt) : LongInt
11641: Function capSetCallbackOnCapControl( hwnd : THandle; fpProc : LongInt) : LongInt
11642: Function capSetUserData( hwnd : THandle; lUser : LongInt) : LongInt
11643: Function capGetUserData( hwnd : THandle) : LongInt
11644: Function capDriverConnect( hwnd : THandle; I : Word) : LongInt
11645: Function capDriverDisconnect( hwnd : THandle) : LongInt
11646: Function capDriverGetName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11647: Function capDriverGetVersion( hwnd : THandle; szVer : LongInt; wSize : Word) : LongInt
11648: Function capDriverGetCaps( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11649: Function capFileSetCaptureFile( hwnd : THandle; szName : LongInt) : LongInt
11650: Function capFileGetCaptureFile( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11651: Function capFileAlloc( hwnd : THandle; dwSize : LongInt) : LongInt
11652: Function capFileSaveAs( hwnd : THandle; szName : LongInt) : LongInt
11653: Function capFileSetInfoChunk( hwnd : THandle; lpInfoChunk : LongInt) : LongInt
11654: Function capFileSaveDIB( hwnd : THandle; szName : LongInt) : LongInt
11655: Function capEditCopy( hwnd : THandle) : LongInt
11656: Function capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11657: Function capGetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11658: Function capGetAudioFormatSize( hwnd : THandle) : LongInt
11659: Function capDlgVideoFormat( hwnd : THandle) : LongInt
11660: Function capDlgVideoSource( hwnd : THandle) : LongInt
11661: Function capDlgVideoDisplay( hwnd : THandle) : LongInt
11662: Function capDlgVideoCompression( hwnd : THandle) : LongInt
11663: Function capGetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11664: Function capGetVideoFormatSize( hwnd : THandle) : LongInt
11665: Function capSetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11666: Function capPreview( hwnd : THandle; f : Word) : LongInt
11667: Function capPreviewRate( hwnd : THandle; wMS : Word) : LongInt
11668: Function capOverlay( hwnd : THandle; f : Word) : LongInt
11669: Function capPreviewScale( hwnd : THandle; f : Word) : LongInt
11670: Function capGetStatus( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11671: Function capSetScrollPos( hwnd : THandle; lpP : LongInt) : LongInt
11672: Function capGrabFrame( hwnd : THandle) : LongInt
11673: Function capGrabFrameNoStop( hwnd : THandle) : LongInt
11674: Function capCaptureSequence( hwnd : THandle) : LongInt
11675: Function capCaptureSequenceNoFile( hwnd : THandle) : LongInt
11676: Function capCaptureStop( hwnd : THandle) : LongInt
11677: Function capCaptureAbort( hwnd : THandle) : LongInt
11678: Function capCaptureSingleFrameOpen( hwnd : THandle) : LongInt
11679: Function capCaptureSingleFrameClose( hwnd : THandle) : LongInt
11680: Function capCaptureSingleFrame( hwnd : THandle) : LongInt
11681: Function capCaptureGetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11682: Function capCaptureSetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11683: Function capSetMCIDeviceName( hwnd : THandle; szName : LongInt) : LongInt
11684: Function capGetMCIDeviceName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11685: Function capPaletteOpen( hwnd : THandle; szName : LongInt) : LongInt
11686: Function capPaletteSave( hwnd : THandle; szName : LongInt) : LongInt
11687: Function capPalettePaste( hwnd : THandle) : LongInt
11688: Function capPaletteAuto( hwnd : THandle; iFrames : Word; iColors : LongInt) : LongInt
11689: Function capPaletteManual( hwnd : THandle; fGrab : Word; iColors : LongInt) : LongInt
11690: //PCapDriverCaps', '^TCapDriverCaps // will not work
11691: TCapDriverCaps', 'record wDeviceIndex : WORD; fHasOverlay : BOOL'
11692: +'; fHasDlgVideoSource : BOOL; fHasDlgVideoFormat : BOOL; fHasDlgVideoDispla'
11693: +'; y : BOOL; fCaptureInitialized : BOOL; fDriverSuppliesPalettes : BOOL; hVid'
11694: +'; eoIn : THANDLE; hVideoOut : THANDLE; hVideoExtIn:THANDLE; hVideoExtOut:THANDLE; end
11695: //PCapStatus', '^TCapStatus // will not work
11696: TCapStatus', 'record uiImageWidth : UINT; uiImageHeight : UINT; '
11697: +'; fLiveWindow : BOOL; fOverlayWindow : BOOL; fScale : BOOL; ptScroll : TPOINT'
11698: +'; fUsingDefaultPalette : BOOL; fAudioHardware : BOOL; fCapFileExists : BO'
11699: +'; OL; dwCurrentVideoFrame : DWORD; dwCurrentVideoFramesDropped : DWORD; dwCu'
11700: +'; rrentWaveSamples : DWORD; dwCurrentTimeElapsedMS : DWORD; hPalCurrent : HP'
11701: +'; ALETTE; fCapturingNow : BOOL; dwReturn : DWORD; wNumVideoAllocated : WORD;'
11702: +'; wNumAudioAllocated : WORD; end
11703: //PCaptureParms', '^TCaptureParms // will not work
11704: TCaptureParms', 'record dwRequestMicroSecPerFrame : DWORD; fMake'
11705: +'; UserHitOKToCapture : BOOL; wPercentDropForError : WORD; fYield : BOOL; dwI'
11706: +'; ndexSize : DWORD; wChunkGranularity : WORD; fUsingDOSMemory : BOOL; wNumVi'
11707: +'; deoRequested : WORD; fCaptureAudio : BOOL; wNumAudioRequested : WORD; vKey'
11708: +'; Abort : WORD; fAbortLeftMouse : BOOL; fAbortRightMouse : BOOL; fLimitEnabl'
11709: +'; ed : BOOL; wTimeLimit : WORD; fMCIControl : BOOL; fStepMCIDevice : BOOL; d'
11710: +'; wMCIStartTime : DWORD; dwMCIStopTime : DWORD; fStepCaptureAt2x : BOOL; wSt'
11711: +'; epCaptureAverageFrames : WORD; dwAudioBufferSize : DWORD; fDisableWriteCac'
11712: +'; he : BOOL; AVStreamMaster : WORD; end
11713: // PCapInfoChunk', '^TCapInfoChunk // will not work
11714: //TCapInfoChunk', 'record fccInfoID : FOURCC; lpData : LongInt; cbData : LongInt; end
11715: 'CONTROLCALLBACK_PREROLL', 'LongInt').SetInt( 1);
11716: 'CONTROLCALLBACK_CAPTURING', 'LongInt').SetInt( 2);
11717: Function capCreateCaptureWindow( lpszWindowName: PChar; dwStyle : DWord; x, y : Integer; nWidth, nHeight
: Integer; hwndParent : THandle; nID : Integer) : THandle
11718: Function
capGetDriverDescription( wDriverIndex: DWord; lpszName: PChar; cbName: Integer; lpszVer: PChar; cbVer: Int): Bool;
11719: 'IDS_CAP_BEGIN', 'LongInt').SetInt( 300);
11720: 'IDS_CAP_END', 'LongInt').SetInt( 301);

```

```

11721: 'IDS_CAP_INFO', 'LongInt').SetInt( 401);
11722: 'IDS_CAP_OUTOFMEM', 'LongInt').SetInt( 402);
11723: 'IDS_CAP_FILEEXISTS', 'LongInt').SetInt( 403);
11724: 'IDS_CAP_ERRORPALOPEN', 'LongInt').SetInt( 404);
11725: 'IDS_CAP_ERRORPALSAVE', 'LongInt').SetInt( 405);
11726: 'IDS_CAP_ERRORDIBSAVE', 'LongInt').SetInt( 406);
11727: 'IDS_CAP_DEFAVIEXT', 'LongInt').SetInt( 407);
11728: 'IDS_CAP_DEFPALEXT', 'LongInt').SetInt( 408);
11729: 'IDS_CAP_CANTOPEN', 'LongInt').SetInt( 409);
11730: 'IDS_CAP_SEQ_MSGSTART', 'LongInt').SetInt( 410);
11731: 'IDS_CAP_SEQ_MSGSTOP', 'LongInt').SetInt( 411);
11732: 'IDS_CAP_VIDEDITERR', 'LongInt').SetInt( 412);
11733: 'IDS_CAP_READONLYFILE', 'LongInt').SetInt( 413);
11734: 'IDS_CAP_WRITEERROR', 'LongInt').SetInt( 414);
11735: 'IDS_CAP_NODISKSPACE', 'LongInt').SetInt( 415);
11736: 'IDS_CAP_SETFILESIZE', 'LongInt').SetInt( 416);
11737: 'IDS_CAP_SAVEASPERCENT', 'LongInt').SetInt( 417);
11738: 'IDS_CAP_DRIVER_ERROR', 'LongInt').SetInt( 418);
11739: 'IDS_CAP_WAVE_OPEN_ERROR', 'LongInt').SetInt( 419);
11740: 'IDS_CAP_WAVE_ALLOC_ERROR', 'LongInt').SetInt( 420);
11741: 'IDS_CAP_WAVE_PREPARE_ERROR', 'LongInt').SetInt( 421);
11742: 'IDS_CAP_WAVE_ADD_ERROR', 'LongInt').SetInt( 422);
11743: 'IDS_CAP_WAVE_SIZE_ERROR', 'LongInt').SetInt( 423);
11744: 'IDS_CAP_VIDEO_OPEN_ERROR', 'LongInt').SetInt( 424);
11745: 'IDS_CAP_VIDEO_ALLOC_ERROR', 'LongInt').SetInt( 425);
11746: 'IDS_CAP_VIDEO_PREPARE_ERROR', 'LongInt').SetInt( 426);
11747: 'IDS_CAP_VIDEO_ADD_ERROR', 'LongInt').SetInt( 427);
11748: 'IDS_CAP_VIDEO_SIZE_ERROR', 'LongInt').SetInt( 428);
11749: 'IDS_CAP_FILE_OPEN_ERROR', 'LongInt').SetInt( 429);
11750: 'IDS_CAP_FILE_WRITE_ERROR', 'LongInt').SetInt( 430);
11751: 'IDS_CAP_RECORDING_ERROR', 'LongInt').SetInt( 431);
11752: 'IDS_CAP_RECORDING_ERROR2', 'LongInt').SetInt( 432);
11753: 'IDS_CAP_AVI_INIT_ERROR', 'LongInt').SetInt( 433);
11754: 'IDS_CAP_NO_FRAME_CAP_ERROR', 'LongInt').SetInt( 434);
11755: 'IDS_CAP_NO_PALETTE_WARN', 'LongInt').SetInt( 435);
11756: 'IDS_CAP_MCI_CONTROL_ERROR', 'LongInt').SetInt( 436);
11757: 'IDS_CAP_MCI_CANT_STEP_ERROR', 'LongInt').SetInt( 437);
11758: 'IDS_CAP_NO_AUDIO_CAP_ERROR', 'LongInt').SetInt( 438);
11759: 'IDS_CAP_AVI_DRAWDIB_ERROR', 'LongInt').SetInt( 439);
11760: 'IDS_CAP_COMPRESSOR_ERROR', 'LongInt').SetInt( 440);
11761: 'IDS_CAP_AUDIO_DROP_ERROR', 'LongInt').SetInt( 441);
11762: 'IDS_CAP_STAT_LIVE_MODE', 'LongInt').SetInt( 500);
11763: 'IDS_CAP_STAT_OVERLAY_MODE', 'LongInt').SetInt( 501);
11764: 'IDS_CAP_STAT_CAP_INIT', 'LongInt').SetInt( 502);
11765: 'IDS_CAP_STAT_CAP_FINI', 'LongInt').SetInt( 503);
11766: 'IDS_CAP_STAT_PALETTE_BUILD', 'LongInt').SetInt( 504);
11767: 'IDS_CAP_STAT_OPTPAL_BUILD', 'LongInt').SetInt( 505);
11768: 'IDS_CAP_STAT_I_FRAMES', 'LongInt').SetInt( 506);
11769: 'IDS_CAP_STAT_L_FRAMES', 'LongInt').SetInt( 507);
11770: 'IDS_CAP_STAT_CAP_L_FRAMES', 'LongInt').SetInt( 508);
11771: 'IDS_CAP_STAT_CAP_AUDIO', 'LongInt').SetInt( 509);
11772: 'IDS_CAP_STAT_VIDEOCURRENT', 'LongInt').SetInt( 510);
11773: 'IDS_CAP_STAT_VIDEOAUDIO', 'LongInt').SetInt( 511);
11774: 'IDS_CAP_STAT_VIDEOONLY', 'LongInt').SetInt( 512);
11775: 'IDS_CAP_STAT_FRAMESDROPPED', 'LongInt').SetInt( 513);
11776: 'AVICAP32', 'String').SetString( 'AVICAP32.dll
11777: end;
11778:
11779: procedure SIRegister_ALFcnMisc(CL: TPSPascalCompiler);
11780: begin
11781:   Function AlBoolToInt( Value : Boolean) : Integer
11782:   Function AlMediumPos( LTotal, LBorder, LObject : integer) : Integer
11783:   Function AlIsValidEmail( const Value : AnsiString) : boolean
11784:   Function AlLocalDateTimeToGMTDateTime( const aLocalDateTime : TDateTime) : TDateTime
11785:   Function AlInc( var x : integer; Count : integer) : Integer
11786:   function AlCopyStr(const aSourceString: AnsiString; aStart, aLength: Integer): AnsiString
11787:   function ALGetStringFromFile(filename: AnsiString; const ShareMode: Word = fmShareDenyWrite):AnsiString;
11788:   procedure ALSaveStringToFile(Str: AnsiString; filename: AnsiString);
11789:   Function ALIsInteger(const S: AnsiString): Boolean;
11790:   function ALIsDecimal(const S: AnsiString): boolean;
11791:   Function ALStringToWideString(const S: AnsiString; const aCodePage: Word): WideString;
11792:   function ALWideStringToString(const WS: WideString; const aCodePage: Word): AnsiString;
11793:   function ALQuotedStr(const S: AnsiString; const Quote: AnsiChar = ''''): AnsiString;
11794:   function ALDequotedStr(const S: AnsiString; AQuote: AnsiChar): AnsiString;
11795:   function ALUTF8removeBOM(const S: AnsiString): AnsiString;
11796:   Function ALRandomStr(const aLength: Longint; const aCharset: Array of Char): AnsiString;
11797:   Function ALRandomStr(const aLength: Longint): AnsiString;
11798:   Function ALRandomStrU1(const aLength: Longint; const aCharset: Array of Char): String;
11799:   Function ALRandomStrU(const aLength: Longint): String;
11800: end;
11801:
11802: procedure SIRegister_ALJSONDoc(CL: TPSPascalCompiler);
11803: begin
11804:   Procedure ALJSONToTStrings(const AJsonStr:AnsiString;aLst:TALStrings; const aNullStr:AnsiString;const
aTrueStr: AnsiString; const aFalseStr : AnsiString)
11805: end;
11806:
11807: procedure SIRegister_ALWindows(CL: TPSPascalCompiler);
11808: begin

```



```

11809: _ALMEMORYSTATUSEx', 'record dwLength : DWORD; dwMemoryLoad : DWO'
11810: + 'RD; ullTotalPhys : Int64; ullAvailPhys : Int64; ullTotalPageFile : Int64; '
11811: + 'ullAvailPageFile : Int64; ullTotalVirtual : Int64; ullAvailVirtual : Int64'
11812: + 'ullAvailExtendedVirtual : Int64; end
11813: TALMemoryStatusEx', '_ALMEMORYSTATUSEx
11814: Function ALGlobalMemoryStatusEx( var lpBuffer : TALMEMORYSTATUSEx ) : BOOL
11815: Function ALInterlockedExchange64( var Target : LONGLONG; Value : LONGLONG ) : LONGLONG
11816: 'INVALID_SET_FILE_POINTER', 'LongInt').SetInt( DWORD ( - 1 ));
11817: 'QUOTA_LIMITS_HARDWS_MIN_DISABLE', 'LongWord').SetUInt( $2);
11818: 'QUOTA_LIMITS_HARDWS_MIN_ENABLE', 'LongWord').SetUInt( $1);
11819: 'QUOTA_LIMITS_HARDWS_MAX_DISABLE', 'LongWord').SetUInt( $8);
11820: 'QUOTA_LIMITS_HARDWS_MAX_ENABLE', 'LongWord').SetUInt( $4);
11821: end;
11822:
11823: procedure SIRegister_IPCThrd(CL: TPSPascalCompiler);
11824: begin
11825:   SIRegister_THandledObject(CL);
11826:   SIRegister_TEvent(CL);
11827:   SIRegister_TMutex(CL);
11828:   SIRegister_TSharedMem(CL);
11829:   'TRACE_BUF_SIZE', 'LongInt').SetInt( 200 * 1024);
11830:   'TRACE_BUFFER', 'String').SetString( 'TRACE_BUFFER
11831:   'TRACE_Mutex', 'String').SetString( 'TRACE_Mutex
11832:   //PTraceEntry', '^TTraceEntry // will not work
11833:   SIRegister_TIPCTracer(CL);
11834:   'MAX_CLIENTS', 'LongInt').SetInt( 6);
11835:   'IPCTIMEOUT', 'LongInt').SetInt( 2000);
11836:   'IPCBUFFER_NAME', 'String').SetString( 'BUFFER_NAME
11837:   'BUFFER_Mutex', 'String').SetString( 'BUFFER_Mutex
11838:   'MONITOR_EVENT_NAME', 'String').SetString( 'MONITOR_EVENT
11839:   'CLIENT_EVENT_NAME', 'String').SetString( 'CLIENT_EVENT
11840:   'CONNECT_EVENT_NAME', 'String').SetString( 'CONNECT_EVENT
11841:   'CLIENT_DIR_NAME', 'String').SetString( 'CLIENT_DIRECTORY
11842:   'CLIENT_DIR_Mutex', 'String').SetString( 'DIRECTORY_Mutex
11843:   FindClass('TOBJECT'), 'EMonitorActive
11844:   FindClass('TOBJECT'), 'TIPCThread
11845:   TEventKind', '( evMonitorAttach, evMonitorDetach, evMonitorSigna'
11846:   + 'l, evMonitorExit, evClientStart, evClientStop, evClientAttach, evClientDet'
11847:   + 'ach, evClientSwitch, evClientSignal, evClientExit )
11848:   TClientFlag', '( cfError, cfMouseMove, cfMouseDown, cfResize, cfAttach )
11849:   TClientFlags', 'set of TClientFlag
11850:   //PEventData', '^TEventData // will not work
11851:   TEventData', 'record X : SmallInt; Y : SmallInt; Flag : TClientF'
11852:   + 'lag; Flags : TClientFlags; end
11853:   TConnectEvent', 'Procedure ( Sender : TIPCThread; Connecting : Boolean)
11854:   TDirUpdateEvent', 'Procedure ( Sender : TIPCThread)
11855:   TIPCTNotifyEvent', 'Procedure ( Sender : TIPCThread; Data : TEventData)
11856:   //PIPCEventInfo', '^TIPCEventInfo // will not work
11857:   TIPCEventInfo', 'record FID: Integer; FKind: TEventKind; FData: TEventData; end
11858:   SIRegister_TIPCEvent(CL);
11859:   //PClientDirRecords', '^TClientDirRecords // will not work
11860:   SIRegister_TClientDirectory(CL);
11861:   TIPCTState', '( stInActive, stDisconnected, stConnected )
11862:   SIRegister_TIPCThread(CL);
11863:   SIRegister_TIPCMonitor(CL);
11864:   SIRegister_TIPCCClient(CL);
11865:   Function IsMonitorRunning( var Hndl : THandle ) : Boolean
11866: end;
11867:
11868: (*-----*)
11869: procedure SIRegister_ALGSMComm(CL: TPSPascalCompiler);
11870: begin
11871:   SIRegister_TALGSMComm(CL);
11872:   Function ALGSMComm_BuildPDUMessage( aSMSCenter, aSMSAddress, aMessage : AnsiString) : AnsiString
11873:   Procedure ALGSMComm_DecodePDUMessage(aPDUMessage:AnsiString;var aSMSCenter,aSMSAddress,
11874:   aMessage:AnsiString);
11875:   Function ALGSMComm_UnicodeToGSM7BitDefaultAlphabet( aMessage : WideString ) : AnsiString
11876:   Function ALGSMComm_GSM7BitDefaultAlphabetToUnicode(aMess:AnsiString;const
11877:   UseGreekAlphabet:Bool):Widestring;
11878:   function ALMatchesMask(const Filename, Mask: AnsiString): Boolean;
11879: end;
11880:
11881: procedure SIRegister_ALHttpCommon(CL: TPSPascalCompiler);
11882: begin
11883:   TALHTTPPropertyChangeEvent', 'Procedure(sender:Tobject;const PropertyIndex:Integer;
11884:   TALHTTPProtocolVersion', '( HTTPpv_1_0, HTTPpv_1_1 )
11885:   TALHTTPMethod', '( HTTPmt_Get,HTTPmt_Post,HTTPmt_Head,HTTPmt_Trace,HTTPmt_Put,HTTPmt_Delete);
11886:   TInternetScheme', 'integer
11887:   TALIPv6Binary', 'array[1..16] of Char;
11888:   // TALIPv6Binary = array[1..16] of ansiChar;
11889:   // TInternetScheme = Integer;
11890:   SIRegister_TALHTTPRequestHeader(CL);
11891:   SIRegister_TALHTTPCookie(CL);
11892:   SIRegister_TALHTTPCookieCollection(CL);
11893:   SIRegister_TALHTTPResponseHeader(CL);
11894:   Function ALHTTPDecode( const AStr : AnsiString ) : AnsiString
11895:   Procedure ALHTTPEncodeParamNameValues( ParamValues : TALStrings)
11896:   // Procedure ALExtractHTTPFields(Separators, WhiteSpace, Quotes:TSysCharSet;
11897:   Content:PAnsiChar;Strings:TALStrings;StripQuotes:Boolean;

```

```

11895: // Procedure ALExtractHeaderFields( Separators,WhiteSpace, Quotes : TSysCharSet; Content : PAnsiChar;
11896: Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11897: // Procedure ALExtractHeaderFieldsWithQuoteEscaped(Separators,WhiteSpace,
11898: Quotes:TSysCharSet;Content:PAnsiChar;Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11899: Function ALRemoveSchemeFromUrl( aUrl : AnsiString) : ansiString
11900: Function ALExtractSchemeFromUrl( aUrl : AnsiString) : TInternetScheme
11901: Function ALExtractHostNameFromUrl( aUrl : AnsiString) : AnsiString
11902: Function ALExtractDomainNameFromUrl( aUrl : AnsiString) : AnsiString
11903: Function ALExtractUrlPathFromUrl( aUrl : AnsiString) : AnsiString
11904: Function ALInternetCrackUrl( aUrl : AnsiString; var SchemeName,HostName,UserName,Password,UrlPath,
11905: ExtraInfo : AnsiString; var PortNumber : integer) : Boolean;
11906: Function ALInternetCrackUrl1( aUrl : AnsiString; var SchemeName, HostName, UserName, Password,UrlPath,
11907: Anchor : AnsiString; Query : TALStrings; var PortNumber : integer) : Boolean;
11908: Function ALInternetCrackUrl2(var Url:AnsiString;var Anchor:AnsiString;Query:TALStrings):Boolean;
11909: Function ALRemoveAnchorFromUrl( aUrl : AnsiString; var aAnchor : AnsiString) : AnsiString;
11910: Function ALRemoveAnchorFromUrl1( aUrl : AnsiString) : AnsiString;
11911: Function ALCombineUrl( RelativeUrl, BaseUrl : AnsiString) : AnsiString;
11912: Function ALCombineUrl1(RelativeUrl, BaseUrl, Anchor : AnsiString; Query:TALStrings) : AnsiString;
11913: Function ALGmtDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
11914: Function ALDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
11915: Function ALTryRfc822StrToGMTDateTime( const S : AnsiString; out Value : TDateTime) : Boolean
11916: Function ALRfc822StrToGMTDateTime( const s : AnsiString) : TDateTime
11917: Function ALTryIPv4StrToNumeric( aIPv4Str : ansiString; var aIPv4Num : Cardinal) : Boolean
11918: Function ALIPv4StrToNumeric( aIPv4 : ansiString) : Cardinal
11919: Function ALNumericToIPv4Str( aIPv4 : Cardinal) : ansiString
11920: Function ALZeroIPv6 : TALIPv6Binary
11921: Function ALTryIPv6StrToBinary( aIPv6Str : ansiString; var aIPv6Bin : TALIPv6Binary) : Boolean
11922: Function ALIPv6StrToBinary( aIPv6 : ansiString) : TALIPv6Binary
11923: Function ALBinaryToIPv6Str( aIPv6 : TALIPv6Binary) : ansiString
11924: Function ALBinaryStrToIPv6Binary( aIPv6BinaryStr : ansiString) : TALIPv6Binary
11925: end;
11926: procedure SIRegister_ALFcnHTML(CL: TPSPascalCompiler);
11927: begin
11928:   Procedure ALUTF8ExtractHTMLText(HtmlCont:AnsiString;LstExtractedResourceText:TALStrings;const
11929:   DecodeHTMLText:Boolean);
11930:   Function ALUTF8ExtractHTMLText1(HtmlContent:AnsiString;const DecodeHTMLText:Boolean): AnsiString;
11931:   Function ALXMLCDataElementEncode( Src : AnsiString) : AnsiString
11932:   Function ALXMLTextElementEncode(Src : AnsiString; const useNumericReference : boolean) : AnsiString
11933:   Function ALUTF8XMLTextElementDecode( const Src : AnsiString) : AnsiString
11934:   Function ALUTF8HTMLEncode(const Src:AnsiStr;const EncodeASCIHtmlEntities:Bool;const
11935:   useNumRef:bool):AnsiString);
11936:   Function ALUTF8HTMLDecode( const Src : AnsiString) : AnsiString
11937:   Function ALJavaScriptEncode( const Src : AnsiString; const useNumericReference : boolean) : AnsiString
11938:   Function ALUTF8JavaScriptDecode( const Src : AnsiString) : AnsiString
11939:   Procedure ALHideHtmlUnwantedTagForHTMLHandleTagfunct(var HtmlContent:AnsiString; const
11940:   DeleteBodyOfUnwantedTag : Boolean; const ReplaceUnwantedTagCharBy : AnsiChar)
11941:   Procedure ALCompactHtmTagParams( TagParams : TALStrings)
11942: end;
11943: procedure SIRegister_ALInternetMessageCommon(CL: TPSPascalCompiler);
11944: begin
11945:   SIRegister_TALEmailHeader(CL);
11946:   SIRegister_TALNewsArticleHeader(CL);
11947:   Function ALParseEmailAddress(FriendlyEmail:AnsiString;var RealName:AString;const
11948:   decodeRealName:Bool):AnsiString;
11949:   Function ALExtractEmailAddress( FriendlyEmail : AnsiString) : AnsiString
11950:   Function ALMakeFriendlyEmailAddress( aRealName, aEmail : AnsiString) : AnsiString
11951:   Function ALEncodeRealName4FriendlyEmailAddress( aRealName : AnsiString) : AnsiString
11952:   Function ALGenerateInternetMessageID : AnsiString;
11953:   Function ALGenerateInternetMessageID1( ahostname : AnsiString) : AnsiString;
11954:   Function ALDecodeQuotedPrintableString( src : AnsiString) : AnsiString
11955:   Function ALDecodeInternetMessageHeaderInUTF8(aHeaderStr:AnsiString;aDefaultCodePage:Integer):AnsiString;
11956: end;
11957: (*-----*)
11958: procedure SIRegister_ALFcnWinSock(CL: TPSPascalCompiler);
11959: begin
11960:   Function ALHostToIP( HostName : AnsiString; var Ip : AnsiString):Boolean
11961:   Function ALIPAddrToName( IPAddr : AnsiString) : AnsiString
11962:   Function ALgetLocalIPs : TALStrings
11963:   Function ALgetLocalHostName : AnsiString
11964: end;
11965: procedure SIRegister_ALFcnCGI(CL: TPSPascalCompiler);
11966: begin
11967:   Procedure ALCGIInitDefaultServerVariablesFromWebRequest(WebRequest :
11968:   TALWebRequest;ServerVariables:TALStrings);
11969:   Procedure ALCGIInitDefaultServerVariablesFromWebRequest1(WebRequest: TALWebRequest; ServerVariables :
11970:   TALStrings; ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
11971:   Procedure ALCGIInitDefaultServerVariables( ServerVariables : TALStrings);
11972:   Procedure ALCGIInitDefaultServerVariables1(ServerVars:TALStrings;ScriptName,
11973:   ScriptFileName:AnsiString;Url:AnsiString);
11974:   Procedure ALCGIInitServerVariablesFromWebRequest( WebRequest:TALWebRequest; ServerVariables : TALStrings;
11975:   ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
11976:   Procedure ALCGIExec( InterpreterFilename : AnsiString; ServerVariables : TALStrings; RequestContentStream
11977:   : Tstream; ResponseContentStream : Tstream; ResponseHeader : TALHTTPResponseHeader);
11978:   Procedure ALCGIExec1(ScriptName,ScriptFileName, Url, X_REWRITE_URL, InterpreterFilename:AnsiString;
11979:   WebRequest : TALIsapiRequest;
11980:   overloadedCookies:AnsiString;overloadedQueryString:AnsiString;overloadedReferer: AnsiString;

```

```

11970:  + 'overloadedRequestContentStream:Tstream;var
ResponseContentStr:AnsiString;ResponseHeader:TALHTTPResponseHeader;
11971:  Procedure ALCGIExec2(ScriptName,ScriptFileName,Url,X_REWRITE_URL,
InterpreterFilename:AnsiString;WebRequest: TALIsapiRequest; var ResponseContentString : AnsiString;
ResponseHeader : TALHTTPResponseHeader);
11972: end;
11973:
11974: procedure SIRegister_ALFcnExecute(CL: TPSPascalCompiler);
11975: begin
11976:   TStartupInfoA, 'TStartupInfo
11977:   'SE_CREATE_TOKEN_NAME','String').SetString( 'SeCreateTokenPrivilege
11978: SE_ASSIGNPRIMARYTOKEN_NAME','String').SetString( 'SeAssignPrimaryTokenPrivilege
11979: SE_LOCK_MEMORY_NAME','String').SetString( 'SeLockMemoryPrivilege
11980: SE_INCREASE_QUOTA_NAME','String').SetString( 'SeIncreaseQuotaPrivilege
11981: SE_UNSOLICITED_INPUT_NAME','String').SetString( 'SeUnsolicitedInputPrivilege
11982: SE_MACHINE_ACCOUNT_NAME','String').SetString( 'SeMachineAccountPrivilege
11983: SE_TCB_NAME','String').SetString( 'SeTcbPrivilege
11984: SE_SECURITY_NAME','String').SetString( 'SeSecurityPrivilege
11985: SE_TAKE_OWNERSHIP_NAME','String').SetString( 'SeTakeOwnershipPrivilege
11986: SE_LOAD_DRIVER_NAME','String').SetString( 'SeLoadDriverPrivilege
11987: SE_SYSTEM_PROFILE_NAME','String').SetString( 'SeSystemProfilePrivilege
11988: SE_SYSTEMTIME_NAME','String').SetString( 'SeSystemtimePrivilege
11989: SE_PROF_SINGLE_PROCESS_NAME','String').SetString( 'SeProfileSingleProcessPrivilege
11990: SE_INC_BASE_PRIORITY_NAME','String').SetString( 'SeIncreaseBasePriorityPrivilege
11991: SE_CREATE_PAGEFILE_NAME','String').SetString( 'SeCreatePagefilePrivilege
11992: SE_CREATE_PERMANENT_NAME','String').SetString( 'SeCreatePermanentPrivilege
11993: SE_BACKUP_NAME','String').SetString( 'SeBackupPrivilege
11994: SE_RESTORE_NAME','String').SetString( 'SeRestorePrivilege
11995: SE_SHUTDOWN_NAME','String').SetString( 'SeShutdownPrivilege
11996: SE_DEBUG_NAME','String').SetString( 'SeDebugPrivilege
11997: SE_AUDIT_NAME','String').SetString( 'SeAuditPrivilege
11998: SE_SYSTEM_ENVIRONMENT_NAME','String').SetString( 'SeSystemEnvironmentPrivilege
11999: SE_CHANGE_NOTIFY_NAME','String').SetString( 'SeChangeNotifyPrivilege
12000: SE_REMOTE_SHUTDOWN_NAME','String').SetString( 'SeRemoteShutdownPrivilege
12001: SE_UNDOCK_NAME','String').SetString( 'SeUndockPrivilege
12002: SE_SYNC_AGENT_NAME','String').SetString( 'SeSyncAgentPrivilege
12003: SE_ENABLE_DELEGATION_NAME','String').SetString( 'SeEnableDelegationPrivilege
12004: SE_MANAGE_VOLUME_NAME','String').SetString( 'SeManageVolumePrivilege
12005: Function ALGetEnvironmentString : AnsiString
12006: Function ALWinExec32(const FileName,CurrentDir,
Environment:AnsiString;InStream:Tstream;OutStream:TStream):Dword;
12007: Function ALWinExec32l(const FileName:AnsiString; InputStream:Tstream;OutputStream:TStream):Dword;
12008: Function ALWinExecAndWait32( FileName : AnsiString; Visibility : integer) : DWORD
12009: Function ALWinExecAndWait32V2( FileName : AnsiString; Visibility : integer) : DWORD
12010: Function ALNTSetPrivilege( sPrivilege : AnsiString; bEnabled : Boolean) : Boolean
12011: end;
12012:
12013: procedure SIRegister_ALFcnFile(CL: TPSPascalCompiler);
12014: begin
12015:   Function ALEmptyDirectory(Directory:ansiString;SubDirectory:Bool;IgnoreFiles:array of AnsiString; const
RemoveEmptySubDirectory : Boolean; const FileNameMask : ansiString; const MinFileAge : TdateTime):Boolean;
12016:   Function ALEmptyDirectoryl( Directory : ansiString; SubDirectory : Boolean; const
RemoveEmptySubDirectory:Bool; const FileNameMask : ansiString; const MinFileAge : TdateTime) : Boolean;
12017:   Function ALCopyDirectory( SrcDirectory, DestDirectory : ansiString; SubDirectory : Boolean; const
FileNameMask : ansiString; const FailIfExists : Boolean) : Boolean
12018:   Function ALGetModuleName : ansistring
12019:   Function ALGetModuleFileNameWithoutExtension : ansistring
12020:   Function ALGetModulePath : ansiString
12021:   Function ALGetFileSize( const AFileName : ansistring) : int64
12022:   Function ALGetFileVersion( const AFileName : ansistring) : ansiString
12023:   Function ALGetFileCreationDateTime( const aFileName : AnsiString) : TDateTime
12024:   Function ALGetFileLastWriteDateTime( const aFileName : AnsiString) : TDateTime
12025:   Function ALGetFileLastAccessDateTime( const aFileName : AnsiString) : TDateTime
12026:   Procedure ALSetFileCreationDateTime( const aFileName : AnsiString; const aCreationDateTime : TDateTime)
12027:   Function ALIsDirectoryEmpty( const directory : ansiString) : boolean
12028:   Function ALFileExists( const Path : ansiString) : boolean
12029:   Function ALDirectoryExists( const Directory : AnsiString) : Boolean
12030:   Function ALCreateDir( const Dir : AnsiString) : Boolean
12031:   Function ALRemoveDir( const Dir : AnsiString) : Boolean
12032:   Function ALDeleteFile( const FileName : AnsiString) : Boolean
12033:   Function ALRenameFile( const OldName, NewName : ansistring) : Boolean
12034: end;
12035:
12036: procedure SIRegister_ALFcnMime(CL: TPSPascalCompiler);
12037: begin
12038:   NativeInt', 'Integer
12039:   NativeUInt', 'Cardinal
12040:   Function ALMimeBase64EncodeString( const S : AnsiString) : AnsiString
12041:   Function ALMimeBase64EncodeStringNoCRLF( const S : AnsiString) : AnsiString
12042:   Function ALMimeBase64DecodeString( const S : AnsiString) : AnsiString
12043:   Function ALMimeBase64EncodedSize( const InputSize : NativeInt) : NativeInt
12044:   Function ALMimeBase64EncodedSizeNoCRLF( const InputSize : NativeInt) : NativeInt
12045:   Function ALMimeBase64DecodedSize( const InputSize : NativeInt) : NativeInt
12046:   Procedure ALMimeBase64Encode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12047:   Procedure ALMimeBase64EncodeNoCRLF( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12048:   Procedure ALMimeBase64EncodeFullLines( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)

```

```

12049: Function ALMimeBase64Decode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt) : NativeInt;
12050: Function ALMimeBase64DecodePartial( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt;
12051: + var ByteBuffer : Cardinal; var ByteBufferSize : Cardinal) : NativeInt;
12052: Function ALMimeBase64DecodePartialEnd( out OutputBuffer : TByteDynArray; OutputOffset : NativeInt; const
ByteBuffer : Cardinal; const ByteBufferSize : Cardinal) : NativeInt;
12053: Procedure ALMimeBase64Encode( const InputBuf : TByteDynArray; const InputByteCnt : NatInt; out
OutputBuf : TByteDynArray);
12054: Procedure ALMimeBase64EncodeNoCRLF( const InputBuffer : TByteDynArray; const InputByteCount : NativeInt; out
OutputBuffer : TByteDynArray);
12055: Procedure ALMimeBase64EncodeFullLines( const InputBuffer : TByteDynArray; const InputByteCount : NativeInt; out
OutputBuffer : TByteDynArray);
12056: Function ALMimeBase64Decode1( const InputBuffer : TByteDynArray; const InputByteCount : NativeInt; out
OutputBuffer : TByteDynArray) : NativeInt;
12057: Function ALMimeBase64DecodePartial1( const InputBuffer : TByteDynArray; const InputByteCount : NativeInt; out
OutputBuffer : TByteDynArray; var ByteBuffer : Cardinal; var ByteBufferSize : Cardinal) : NativeInt;
12058: Function ALMimeBase64DecodePartialEnd1( out OutputBuffer : TByteDynArray; const ByteBuffer : Cardinal; const
ByteBufferSpace : Cardinal) : NativeInt;
12059: Procedure ALMimeBase64EncodeFile( const InputFileName, OutputFileName : TFileName)
12060: Procedure ALMimeBase64EncodeFileNoCRLF( const InputFileName, OutputFileName : TFileName)
12061: Procedure ALMimeBase64DecodeFile( const InputFileName, OutputFileName : TFileName)
12062: Procedure ALMimeBase64EncodeStream( const InputStream : TStream; const OutputStream : TStream)
12063: Procedure ALMimeBase64EncodeStreamNoCRLF( const InputStream : TStream; const OutputStream : TStream)
12064: Procedure ALMimeBase64DecodeStream( const InputStream : TStream; const OutputStream : TStream)
12065: 'cALMimeBase64_ENCODED_LINE_BREAK', 'LongInt').SetInt( 76);
12066: 'cALMimeBase64_DECODED_LINE_BREAK', 'LongInt').SetInt( cALMimeBase64_ENCODED_LINE_BREAK div 4 * 3);
12067: 'cALMimeBase64_BUFFER_SIZE', 'LongInt').SetInt( cALMimeBase64_DECODED_LINE_BREAK * 3 * 4 * 4);
12068: Procedure ALFillMimeContentTypeByExtList( AMIMEList : TALStrings)
12069: Procedure ALFillExtByMimeContentTypeList( AMIMEList : TALStrings)
12070: Function ALGetDefaultFileExtFromMimeContentType( aContentType : AnsiString) : AnsiString
12071: Function ALGetDefaultMIMEContentTypeFromExt( aExt : AnsiString) : AnsiString
12072: end;
12073:
12074: procedure SIRegister_ALXmlDoc(CL: TPSPascalCompiler);
12075: begin
12076: 'cALXMLNodeMaxListSize', 'LongInt').SetInt( Maxint div 16);
12077: FindClass('TOBJECT'), 'TALXMLNode
12078: FindClass('TOBJECT'), 'TALXMLNodeList
12079: FindClass('TOBJECT'), 'TALXMLDocument
12080: TALXMLParseProcessingInstructionEvent', 'Procedure (Sender:TObject; const Target,Data:AnsiString)
12081: TALXMLParseTextEvent', 'Procedure ( Sender : TObject; const str: AnsiString)
12082: TALXMLParseStartElementEvent', 'Procedure ( Sender : TObject; co
12083: + 'nt Name : AnsiString; const Attributes : TALStrings)
12084: TALXMLParseEndElementEvent', 'Procedure ( Sender : TObject; const Name : AnsiString)
12085: TALXmlNodeType', '( ntReserved, ntElement, ntAttribute, ntText, '
12086: + 'ntCDATA, ntEntityRef, ntEntity, ntProcessingInstr, ntComment, ntDocument, '
12087: + 'ntDocType, ntDocFragment, ntNotation )
12088: TALXMLDocOption', '( doNodeAutoCreate, doNodeAutoIndent )
12089: TALXMLDocOptions', 'set of TALXMLDocOption
12090: TALXMLParseOption', '( poPreserveWhiteSpace, poIgnoreXMLReferences )
12091: TALXMLParseOptions', 'set of TALXMLParseOption
12092: TALXMLPrologItem', '( xpVersion, xpEncoding, xpStandalone )
12093: PALPointerXMLNodeList', '^TALPointerXMLNodeList // will not work
12094: SIRegister_EALXMLDocError(CL);
12095: SIRegister_TALXMLNodeList(CL);
12096: SIRegister_TALXMLNode(CL);
12097: SIRegister_TALXmlElementNode(CL);
12098: SIRegister_TALXmlAttributeNode(CL);
12099: SIRegister_TALXmlTextNode(CL);
12100: SIRegister_TALXmlDocumentNode(CL);
12101: SIRegister_TALXmlCommentNode(CL);
12102: SIRegister_TALXmlProcessingInstrNode(CL);
12103: SIRegister_TALXmlCDATANode(CL);
12104: SIRegister_TALXmlEntityRefNode(CL);
12105: SIRegister_TALXmlEntityNode(CL);
12106: SIRegister_TALXmlDocTypeNode(CL);
12107: SIRegister_TALXmlDocFragmentNode(CL);
12108: SIRegister_TALXmlNotationNode(CL);
12109: SIRegister_TALXMLDocument(CL);
12110: cALXMLUTF8EncodingStr, 'String').SetString( 'UTF-8
12111: cALXMLUTF8HeaderStr, 'String').SetString('<?xmlversion="1.0"encoding="UTF-8"standalone="yes"?>'+#13#10);
12112: CALNSDelim, 'String').SetString( '
12113: CALXML, 'String').SetString( 'xml
12114: CALVersion, 'String').SetString( 'version
12115: CALEncoding, 'String').SetString( 'encoding
12116: CALStandalone, 'String').SetString( 'standalone
12117: CALDefaultNodeIndent, 'String').SetString( '
12118: CALXmlDocument, 'String').SetString( 'DOCUMENT
12119: Function ALCreateEmptyXMLDocument( const Rootname : AnsiString) : TalXMLDocument
12120: Procedure ALClearXMLDocument( const rootname:AnsiString;xmlDoc:TalXMLDocument;const
EncodingStr:AnsiString);
12121: Function ALFindXmlNodeByChildNodeValue(xmlrec:TalxmlNode;const ChildNodeName,
ChildNodeValue:AnsiString;const Recurse: Boolean) : TalxmlNode
12122: Function ALFindXmlNodeByNameAndChildNodeValue( xmlrec : TalxmlNode; const NodeName : ansiString; const
ChildNodeName, ChildNodeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12123: Function ALFindXmlNodeByAttribute(xmlrec:TalxmlNode;const AttributeName,AttributeValue:AnsiString;const
Recurse: Boolean):TalxmlNode
12124: Function ALFindXmlNodeByNameAndAttribute( xmlrec : TalxmlNode; const NodeName : ansiString; const
AttributeName, AttributeValue : AnsiString; const Recurse : Boolean) : TalxmlNode

```



```

12125:  Function ALExtractAttrValue( const AttrName, AttrLine : AnsiString; const Default : AnsiString) :
        AnsiString
12126:  end;
12127:
12128:  procedure SIRegister_TeCanvas(CL: TPSPascalCompiler);
12129:  //based on TEEProc, TeCanvas, TEEEngine, TChart
12130:  begin
12131:    'TeePiStep','Double').setExtended( Pi / 180.0);
12132:    'TeeDefaultPerspective','LongInt').SetInt( 100);
12133:    'TeeMinAngle','LongInt').SetInt( 270);
12134:    'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12135:    'teeclSkyBlue','LongWord').SetUInt( TColor ( $FOCAA6 ));
12136:    'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ));
12137:    'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12138:    'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12139:    'teeclSkyBlue','LongWord').SetUInt( TColor ( $FOCAA6 ));
12140:    'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ));
12141:    'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12142:    'TA_LEFT','LongInt').SetInt( 0);
12143:    'TA_RIGHT','LongInt').SetInt( 2);
12144:    'TA_CENTER','LongInt').SetInt( 6);
12145:    'TA_TOP','LongInt').SetInt( 0);
12146:    'TA_BOTTOM','LongInt').SetInt( 8);
12147:    'teePATCOPY','LongInt').SetInt( 0);
12148:    'NumCirclePoints','LongInt').SetInt( 64);
12149:    'teeDEFAULT_CHARSET','LongInt').SetInt( 1);
12150:    'teeANTIALIASED_QUALITY','LongInt').SetInt( 4);
12151:    'TA_LEFT','LongInt').SetInt( 0);
12152:    'bs_Solid','LongInt').SetInt( 0);
12153:    'teepf24Bit','LongInt').SetInt( 0);
12154:    'teepfDevice','LongInt').SetInt( 1);
12155:    'CM_MOUSELEAVE','LongInt').SetInt( 10000);
12156:    'CM_SYSCOLORCHANGE','LongInt').SetInt( 10001);
12157:    'DC_BRUSH','LongInt').SetInt( 18);
12158:    'DC_PEN','LongInt').SetInt( 19);
12159:    teeCOLORREF, 'LongWord
12160:    TLogBrush', 'record lbStyle : Integer; lbColor : TColor; lbHatch: Integer; end
12161:    //TNotifyEvent', 'Procedure ( Sender : TObject)
12162:    SIRegister_TFilterRegion(CL);
12163:    SIRegister_IFormCreator(CL);
12164:    SIRegister_TTeeFilter(CL);
12165:    //TFilterClass', 'class of TTeeFilter
12166:    SIRegister_TFilterItems(CL);
12167:    SIRegister_TConvolveFilter(CL);
12168:    SIRegister_TBlurFilter(CL);
12169:    SIRegister_TTeePicture(CL);
12170:    TPenEndStyle', '( esRound, esSquare, esFlat )
12171:    SIRegister_TChartPen(CL);
12172:    SIRegister_TChartHiddenPen(CL);
12173:    SIRegister_TDottedGrayPen(CL);
12174:    SIRegister_TDarkGrayPen(CL);
12175:    SIRegister_TWhitePen(CL);
12176:    SIRegister_TChartBrush(CL);
12177:    TTeeView3DScrolled', 'Procedure ( IsHoriz : Boolean)
12178:    TTeeView3DChangedZoom', 'Procedure ( NewZoom : Integer)
12179:    SIRegister_TView3DOptions(CL);
12180:    FindClass('TOBJECT'),'TTeeCanvas
12181:    TTeeTransparency', 'Integer
12182:    SIRegister_TTeeBlend(CL);
12183:    FindClass('TOBJECT'),'TCanvas3D
12184:    SIRegister_TTeeShadow(CL);
12185:    teeTGradientDirection', '( gdTopBottom, gdBottomTop, gdLeftRight, g'
12186:    + 'dRightLeft, gdFromCenter, gdFromTopLeft, gdFromBottomLeft, gdRadial,gdDiagonalUp,gdDiagonalDown )
12187:    FindClass('TOBJECT'),'TSubGradient
12188:    SIRegister_TCustomTeeGradient(CL);
12189:    SIRegister_TSubGradient(CL);
12190:    SIRegister_TTeeGradient(CL);
12191:    SIRegister_TTeeFontGradient(CL);
12192:    SIRegister_TTeeFont(CL);
12193:    TCanvasBackMode', '( cbmNone, cbmTransparent, cbmOpaque )
12194:    TCanvasTextAlign', 'Integer
12195:    TTeeCanvasHandle', 'HDC
12196:    SIRegister_TTeeCanvas(CL);
12197:    TPoint3DFloat', 'record X : Double; Y : Double; Z : Double; end
12198:    SIRegister_TFloatXYZ(CL);
12199:    TPoint3D', 'record x : integer; y : integer; z : Integer; end
12200:    TRGB', 'record blue: byte; green: byte; red: byte; end
12201:    {TRGB=packed record
12202:      Blue : Byte;
12203:      Green : Byte;
12204:      Red : Byte;
12205:    //IFDEF CLX //Alpha : Byte; // Linux end;}
12206:
12207:    TTeeCanvasCalcPoints', 'Function ( x, z : Integer; var P0, P1 : '
12208:      + 'TPoint3D; var Color0, Color1 : TColor) : Boolean
12209:    TTeeCanvasSurfaceStyle', '( tcsSolid, tcsWire, tcsDot )
12210:    TCanvas3DPlane', '( cpX, cpY, cpZ )
12211:    //IInterface', 'IUnknown
12212:    SIRegister_TCanvas3D(CL);

```

```

12213: SIRegister_TTeeCanvas3D(CL);
12214: TTrianglePoints', 'Array[0..2] of TPoint;
12215: TFourPoints', 'Array[0..3] of TPoint;
12216: Function ApplyDark( Color : TColor; HowMuch : Byte) : TColor
12217: Function ApplyBright( Color : TColor; HowMuch : Byte) : TColor
12218: Function Point3D( const x, y, z : Integer) : TPoint3D
12219: Procedure SwapDouble( var a, b : Double)
12220: Procedure SwapInteger( var a, b : Integer)
12221: Procedure RectSize( const R : TRect; var RectWidth, RectHeight : Integer)
12222: Procedure teeRectCenter( const R : TRect; var X, Y : Integer)
12223: Function RectFromPolygon( const Points : array of TPoint; NumPoints : Integer) : TRect
12224: Function RectFromTriangle( const Points : TTrianglePoints) : TRect
12225: Function RectangleInRectangle( const Small, Big : TRect) : Boolean
12226: Procedure ClipCanvas( ACanvas : TCanvas; const Rect : TRect)
12227: Procedure UnClipCanvas( ACanvas : TCanvas)
12228: Procedure ClipEllipse( ACanvas : TTeeCanvas; const Rect : TRect)
12229: Procedure ClipRoundRectangle(ACanvas:TTeeCanvas;const Rect : TRect; RoundSize : Integer)
12230: Procedure ClipPolygon(ACanvas:TTeeCanvas;const Points:array of TPoint;NumPoints:Integer)
12231: 'TeeCharForHeight','String').SetString( 'W
12232: 'DarkerColorQuantity','Byte').SetUInt( 128);
12233: 'DarkColorQuantity','Byte').SetUInt( 64);
12234: TButtonGetColorProc', 'Function : TColor
12235: SIRegister_TTeeButton(CL);
12236: SIRegister_TButtonColor(CL);
12237: SIRegister_TComboFlat(CL);
12238: Procedure TeeSetTeePen(FPen:TPen; APen : TChartPen; AColor : TColor; Handle:TTeeCanvasHandle)
12239: Function TeePoint( const aX, aY : Integer) : TPoint
12240: Function TEPointInRect( const Rect : TRect; const P : TPoint) : Boolean;
12241: Function PointInRect1( const Rect : TRect; x, y : Integer) : Boolean;
12242: Function TeeRect( Left, Top, Right, Bottom : Integer) : TRect
12243: Function OrientRectangle( const R : TRect) : TRect
12244: Procedure TeeSetBitmapSize( Bitmap : TBitmap; Width, Height : Integer)
12245: Function PolygonBounds( const P : array of TPoint) : TRect
12246: Function PolygonInPolygon( const A, B : array of TPoint) : Boolean
12247: Function RGBValue( const Color : TColor) : TRGB
12248: Function EditColor( AOwner : TComponent; AColor : TColor) : TColor
12249: Function EditColorDialog( AOwner : TComponent; var AColor : TColor) : Boolean
12250: Function PointAtDistance( AFrom, ATo : TPoint; ADist : Integer) : TPoint
12251: Function TeeCull( const P : TFourPoints) : Boolean;
12252: Function TeeCull1( const P0, P1, P2 : TPoint) : Boolean;
12253: TSmoothStretchOption', '( ssBestQuality, ssBestPerformance )
12254: Procedure SmoothStretch( Src, Dst : TBitmap);
12255: Procedure SmoothStretch1( Src, Dst : TBitmap; Option : TSmoothStretchOption);
12256: Function TeeDistance( const x, y : Double) : Double
12257: Function TeeLoadLibrary( const FileName : String) : HInst
12258: Procedure TeeFreeLibrary( hLibModule : HMODULE)
12259: Procedure TeeBlendBitmaps( const Percent : Double; ABitmap, BBitmap : TBitmap; BOrigin : TPoint)
12260: //Procedure TeeCalcLines( var Lines : TRGBArray; Bitmap : TBitmap)
12261: Procedure TeeShadowSmooth(Bitmap, Back : TBitmap; Left, Top, Width, Height, horz, vert : Integer;
Smoothness : Double; FullDraw : Boolean; ACanvas : TCanvas3D; Clip : Boolean)
12262: SIRegister_ICanvasHyperlinks(CL);
12263: SIRegister_ICanvasToolTips(CL);
12264: Function Supports( const Instance : IInterface; const IID : TGUID) : Boolean
12265: end;
12266:
12267: procedure SIRegister_ovcmisc(CL: TPSPascalCompiler);
12268: begin
12269:   TOvcHdc', 'Integer
12270:   TOvcHWND', 'Cardinal
12271:   TOvcHdc', 'HDC
12272:   TOvcHWND', 'HWND
12273:   Function LoadBaseBitmap( lpBitmapName : PChar) : HBITMAP
12274:   Function LoadBaseCursor( lpCursorName : PChar) : HCURSOR
12275:   Function ovCompStruct( const S1, S2, Size : Cardinal) : Integer
12276:   Function DefaultEpoch : Integer
12277:   Function DrawButtonFrame(Canvas:TCanvas;const Client:TRect;IsDown,IsFlat:Bool;Style:TButtonStyle):TRect;
12278:   Procedure FixRealPrim( P : PChar; DC : Char)
12279:   Function GetDisplayString( Canvas : TCanvas; const S : string; MinChars, MaxWidth : Integer) : string
12280:   Function GetLeftButton : Byte
12281:   Function GetNextDlgItem( Ctrl : TOvcHWND) : hWnd
12282:   Procedure GetRGB( Clr : TColor; var IR, IG, IB : Byte)
12283:   Function GetShiftFlags : Byte
12284:   Function ovCreateRotatedFont( F : TFont; Angle : Integer) : hFont
12285:   Function GetTopTextMargin(Font:TFont;BorderStyle:TBorderStyle; Height:Integer;Ctl3D:Boolean): Integer
12286:   Function ovExtractWord( N : Integer; const S : string; WordDelims : TCharSet) : string
12287:   Function ovIsForegroundTask : Boolean
12288:   Function ovTrimLeft( const S : string) : string
12289:   Function ovTrimRight( const S : string) : string
12290:   Function ovQuotedStr( const S : string) : string
12291:   Function ovWordCount( const S : string; const WordDelims : TCharSet) : Integer
12292:   Function ovWordPosition(const N:Integer;const S: string;const WordDelims : TCharSet) : Integer
12293:   Function PtrDiff( const P1, P2 : PChar) : Word
12294:   Procedure PtrInc( var P, Delta : Word)
12295:   Procedure PtrDec( var P, Delta : Word)
12296:   Procedure FixTextBuffer( InBuf, OutBuf : PChar; OutSize : Integer)
12297:   Procedure TransStretchBlt( DstDC : TOvcHdc; DstX, DstY, DstW, DstH : Integer; SrcDC : TOvcHdc; SrcX, SrcY,
SrcW, SrcH : Integer; MaskDC : TOvcHdc; MaskX, MaskY : Integer)
12298:   Function ovMinI( X, Y : Integer) : Integer
12299:   Function ovMaxI( X, Y : Integer) : Integer

```

```

12300: Function ovMinL( X, Y : LongInt) : LongInt
12301: Function ovMaxL( X, Y : LongInt) : LongInt
12302: Function GenerateComponentName( PF : TwinControl; const Root : string) : string
12303: Function PartialCompare( const S1, S2 : string) : Boolean
12304: Function PathEllipsis( const S : string; MaxWidth : Integer) : string
12305: Function ovCreateDisabledBitmap( FOriginal : TBitmap; OutlineColor : TColor) : TBitmap
12306: Procedure ovCopyParentImage( Control : TControl; Dest : TCanvas)
12307: Procedure ovDrawTransparentBitmap( Dest : TCanvas; X, Y, W, H : Integer; Rect : TRect; Bitmap : TBitmap;
TransparentColor : TColor)
12308: Procedure DrawTransparentBitmapPrim(DC:TovcHdc;Bitmap:HBitmap;xStart,yStart,Width,Height:Int;Rect:TRect;
TransparentColor : TColorRef)
12309: Function ovWidthOf( const R : TRect) : Integer
12310: Function ovHeightOf( const R : TRect) : Integer
12311: Procedure ovDebugOutput( const S : string)
12312: Function GetArrowWidth( Width, Height : Integer) : Integer
12313: Procedure StripCharSeq( CharSeq : string; var Str : string)
12314: Procedure StripCharFromEnd( aChr : Char; var Str : string)
12315: Procedure StripCharFromFront( aChr : Char; var Str : string)
12316: Function SystemParametersInfo( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12317: Function SystemParametersInfoNC(uiAction,uiParam:UINT;pvParam:TNonClientMetrics;fWinIni:UINT):BOOL;
12318: Function SystemParametersInfoA( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12319: Function CreateEllipticRgn( p1, p2, p3, p4 : Integer) : HRGN
12320: Function CreateEllipticRgnIndirect( const p1 : TRect) : HRGN
12321: Function CreateFontIndirect( const p1 : TLogFont) : HFONT
12322: Function CreateMetaFile( p1 : PChar) : HDC
12323: Function DescribePixelFormat(DC: HDC;p2:Int;p3:UINT;var p4:TPixelFormatDescriptor): BOOL
12324: Function DrawText(hDC:HDC;lpString:PChar;nCount:Integer; var lpRect : TRect;uFormat:UINT):Integer
12325: Function DrawTextS(hDC:HDC;lpString:string;nCount:Integer; var lpRect:TRect;uFormat:UINT):Integer
12326: Function SetMapperFlags( DC : HDC; Flag : DWORD) : DWORD
12327: Function SetGraphicsMode( hdc : HDC; iMode : Integer) : Integer
12328: Function SetMapMode( DC : HDC; p2 : Integer) : Integer
12329: Function SetMetaFileBitsEx( Size : UINT; const Data : PChar) : HMETAFILE
12330: //Function SetPaletteEntries(Palette:HPALETTE;StartIndex,NumEntries:UINT;var PaletteEntries):UINT
12331: Function SetPixel( DC : HDC; X, Y : Integer; Color : COLORREF) : COLORREF
12332: Function SetPixelV( DC : HDC; X, Y : Integer; Color : COLORREF) : BOOL
12333: //Function SetPixelFormat( DC : HDC; PixelFormat : Integer; FormatDef : PPixelFormatDescriptor) : BOOL
12334: Function SetPolyFillMode( DC : HDC; PolyFillMode : Integer) : Integer
12335: Function StretchBlt(DestDC:HDC;X,Y,Width,Height:Int;SrcDC:HDC;XSrc,YSrc,SrcWidth,
SrcHeight:Int;Rop:DWORD):BOOL
12336: Function SetRectRgn( Rgn : HRgn; X1, Y1, X2, Y2 : Integer) : BOOL
12337: Function StretchDIBits(DC : HDC; DestX, DestY, DestWidth, DestHeight, SrcX, SrcY, SrcWidth,
SrcHeight:Int;Bits:int; var BitsInfo : TBitmapInfo; Usage : UINT; Rop : DWORD) : Integer
12338: Function SetROP2( DC : HDC; p2 : Integer) : Integer
12339: Function SetStretchBltMode( DC : HDC; StretchMode : Integer) : Integer
12340: Function SetSystemPaletteUse( DC : HDC; p2 : UINT) : UINT
12341: Function SetTextCharacterExtra( DC : HDC; CharExtra : Integer) : Integer
12342: Function SetTextColor( DC : HDC; Color : COLORREF) : COLORREF
12343: Function SetTextAlign( DC : HDC; Flags : UINT) : UINT
12344: Function SetTextJustification( DC : HDC; BreakExtra, BreakCount : Integer) : Integer
12345: Function UpdateColors( DC : HDC) : BOOL
12346: Function GetViewportExtEx( DC : HDC; var Size : TSize) : BOOL
12347: Function GetViewportOrgEx( DC : HDC; var Point : TPoint) : BOOL
12348: Function GetWindowExtEx( DC : HDC; var Size : TSize) : BOOL
12349: Function GetWindowOrgEx( DC : HDC; var Point : TPoint) : BOOL
12350: Function IntersectClipRect( DC : HDC; X1, Y1, X2, Y2 : Integer) : Integer
12351: Function InvertRgn( DC : HDC; p2 : HRGN) : BOOL
12352: Function MaskBlt( DestDC : HDC; XDest, YDest, Width, Height : Integer; SrcDC : HDC; XSrc, YSrc : Integer;
Mask : HBITMAP; xMask, yMask : Integer; Rop : DWORD) : BOOL
12353: Function PlgBlt(DestDC:HDC;const PtsArray,SrcDC:HDC;XSrc,YSrc,Widt,Heigh:Int;Mask:HBITMAP;xMask,
yMask:Int):BOOL;
12354: Function OffsetClipRgn( DC : HDC; XOffset, YOffset : Integer) : Integer
12355: Function OffsetRgn( RGN : HRGN; XOffset, YOffset : Integer) : Integer
12356: Function PatBlt( DC : HDC; X, Y, Width, Height : Integer; Rop : DWORD) : BOOL
12357: Function Pie( DC : HDC; X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Integer) : BOOL
12358: Function PlayMetaFile( DC : HDC; MF : HMETAFILE) : BOOL
12359: Function PaintRgn( DC : HDC; RGN : HRGN) : BOOL
12360: Function PtInRegion( RGN : HRGN; X, Y : Integer) : BOOL
12361: Function PtVisible( DC : HDC; X, Y : Integer) : BOOL
12362: Function RectInRegion( RGN : HRGN; const Rect : TRect) : BOOL
12363: Function RectVisible( DC : HDC; const Rect : TRect) : BOOL
12364: Function Rectangle( DC : HDC; X1, Y1, X2, Y2 : Integer) : BOOL
12365: Function RestoreDC( DC : HDC; SavedDC : Integer) : BOOL
12366: end;
12367:
12368: procedure SIRegister_ovcfile(CL: TPSPascalCompiler);
12369: begin
12370:   SIRegister_TovcAbstractStore(CL);
12371:   //PEXPropInfo', '^TExPropInfo // will not work
12372:   // TExPropInfo', 'record PI : TPropInfo; AObject : TObject; end
12373:   SIRegister_TovcPropertyList(CL);
12374:   SIRegister_TovcDataFiler(CL);
12375: Procedure UpdateStoredList( AForm : TwinControl; AStoredList : TStrings; FromForm : Boolean)
12376: Procedure UpdateStoredList1( AForm : TCustomForm; AStoredList : TStrings; FromForm : Boolean)
12377: Function CreateStoredItem( const CompName, PropName : string) : string
12378: Function ParseStoredItem( const Item : string; var CompName, PropName : string) : Boolean
12379: //Function GetPropType( PropInfo : PExPropInfo) : PTypeInfo
12380: end;
12381:
12382: procedure SIRegister_ovccoco(CL: TPSPascalCompiler);

```

```

12383: begin
12384:   'ovsetsize','LongInt').SetInt( 16);
12385:   'etSyntax','LongInt').SetInt( 0);
12386:   'etSymantic','LongInt').SetInt( 1);
12387:   'chCR','Char').SetString( #13);
12388:   'chLF','Char').SetString( #10);
12389:   'chLineSeparator','').SetString( chCR);
12390:   SIRegister_TCocoError(CL);
12391:   SIRegister_TCommentItem(CL);
12392:   SIRegister_TCommentList(CL);
12393:   SIRegister_TSymbolPosition(CL);
12394: TGenListType', '( glNever, glAlways, glOnError )
12395: TovBitSet', 'set of Integer
12396:   //PStartTable', '^TStartTable // will not work
12397: 'TovCharSet', 'set of AnsiChar
12398: TAfterGenListEvent', 'Procedure ( Sender : TObject; var PrintErrorCount : boolean)
12399: TCommentEvent', 'Procedure ( Sender : TObject; CommentList : TCommentList)
12400: TCustomErrorEvent', 'Function(Sender:TObject;const ErrorCode: longint;const Data:string): string
12401: TovErrorEvent', 'Procedure ( Sender : TObject; Error : TCocoError)
12402: TovErrorProc', 'Procedure ( ErrorCode : integer; Symbol : TSymbolP
12403:   +osition; const Data : string; ErrorType : integer)
12404: TFailureEvent', 'Procedure ( Sender : TObject; NumErrors : integer)
12405: TGetCH', 'Function ( pos : longint) : char
12406: TStatusUpdateProc', 'Procedure ( Sender : TObject; Status: string; LineNum:integer)
12407:   SIRegister_TCocoRScanner(CL);
12408:   SIRegister_TCocoRGrammar(CL);
12409:   '_EF','Char').SetString( #0);
12410:   '_TAB','Char').SetString( #09);
12411:   '_CR','Char').SetString( #13);
12412:   '_LF','Char').SetString( #10);
12413:   '_EL','').SetString( _CR);
12414:   '_EOF','Char').SetString( #26);
12415:   'LineEnds','TCharSet').SetInt(ord(_CR) or ord(_LF) or ord(_EF));
12416:   'minErrDist','LongInt').SetInt( 2);
12417:   Function ovPadL( S : string; ch : char; L : integer) : string
12418: end;
12419:
12420:   TFormatSettings = record
12421:     CurrencyFormat: Byte;
12422:     NegCurrFormat: Byte;
12423:     ThousandSeparator: Char;
12424:     DecimalSeparator: Char;
12425:     CurrencyDecimals: Byte;
12426:     DateSeparator: Char;
12427:     TimeSeparator: Char;
12428:     ListSeparator: Char;
12429:     CurrencyString: string;
12430:     ShortDateFormat: string;
12431:     LongDateFormat: string;
12432:     TimeAMString: string;
12433:     TimePMString: string;
12434:     ShortTimeFormat: string;
12435:     LongTimeFormat: string;
12436:     ShortMonthNames: array[1..12] of string;
12437:     LongMonthNames: array[1..12] of string;
12438:     ShortDayNames: array[1..7] of string;
12439:     LongDayNames: array[1..7] of string;
12440:     TwoDigitYearCenturyWindow: Word;
12441:   end;
12442:
12443: procedure SIRegister_OvcFormatSettings(CL: TPSPascalCompiler);
12444: begin
12445:   Function ovFormatSettings : TFormatSettings
12446: end;
12447:
12448: procedure SIRegister_ovcstr(CL: TPSPascalCompiler);
12449: begin
12450:   TOvcCharSet', 'set of Char
12451:   ovBTable', 'array[0..255] of Byte
12452:   //BTable = array[0..{$IFDEF UNICODE}{$IFDEF
HUGE_UNICODE_BMTABLE}$FFFF{$ELSE}$FF{$ENDIF}{$ELSE}$FF{$ENDIF}] of Byte;
12453:   Function BinaryBPChar( Dest : PChar; B : Byte) : PChar
12454:   Function BinaryLPChar( Dest : PChar; L : LongInt) : PChar
12455:   Function BinaryWPChar( Dest : PChar; W : Word) : PChar
12456:   Procedure BMMakeTable( MatchString : PChar; var BT : ovBTable)
12457:   Function BMSearch(var Buffer,BufLength:Cardinal;var BT:ovBTable; MatchString:PChar;var Pos:Cardinal):Bool;
12458:   Function BMSearchUC(var Buffer,BufLength:Card; var BT:ovBTable;MatchString:PChar; var Pos: Card):Boolean
12459:   Function CharStrPChar( Dest : PChar; C : Char; Len : Cardinal) : PChar
12460:   Function DetabPChar( Dest : PChar; Src : PChar; TabSize : Byte) : PChar
12461:   Function HexBPChar( Dest : PChar; B : Byte) : PChar
12462:   Function HexLPChar( Dest : PChar; L : LongInt) : PChar
12463:   Function HexPtrPChar( Dest : PChar; P : TObject) : PChar
12464:   Function HexWPChar( Dest : PChar; W : Word) : PChar
12465:   Function LoCaseChar( C : Char) : Char
12466:   Function OctallLPChar( Dest : PChar; L : LongInt) : PChar
12467:   Function StrChDeletePrim( P : PChar; Pos : Cardinal) : PChar
12468:   Function StrChInsertPrim( Dest : PChar; C : Char; Pos: Cardinal) : PChar
12469:   Function StrChPos( P : PChar; C : Char; var Pos : Cardinal) : Boolean
12470:   Procedure StrInsertChars( Dest : PChar; Ch : Char; Pos, Count : Word)

```



```

12471: Function StrStCopy( Dest, S : PChar; Pos, Count : Cardinal) : PChar
12472: Function StrStDeletePrim( P : PChar; Pos, Count : Cardinal) : PChar
12473: Function StrStInsert( Dest, S1, S2 : PChar; Pos : Cardinal) : PChar
12474: Function StrStInsertPrim( Dest, S : PChar; Pos : Cardinal) : PChar
12475: Function StrStPos( P, S : PChar; var Pos : Cardinal) : Boolean
12476: Function StrToLongPChar( S : PChar; var I : LongInt) : Boolean
12477: Procedure TrimAllSpacesPChar( P : PChar)
12478: Function TrimEmbeddedZeros( const S : string) : string
12479: Procedure TrimEmbeddedZerosPChar( P : PChar)
12480: Function TrimTrailPrimPChar( S : PChar) : PChar
12481: Function TrimTrailPChar( Dest, S : PChar) : PChar
12482: Function TrimTrailingZeros( const S : string) : string
12483: Procedure TrimTrailingZerosPChar( P : PChar)
12484: Function UpCaseChar( C : Char) : Char
12485: Function ovcCharInSet( C : Char; const CharSet : TOvcCharSet) : Boolean
12486: Function ovc32StringIsCurrentCodePage( const S : WideString) : Boolean;
12487: //Function ovc32StringIsCurrentCodePage1( const S:PWideChar; CP : Cardinal) : Boolean;
12488: end;
12489:
12490: procedure SIRegister_AfUtils(CL: TPSPascalCompiler);
12491: begin
12492:   //PraiseFrame, '^TRaiseFrame // will not work
12493:   TRaiseFrame', 'record NextRaise : PRaiseFrame; ExceptAddr : __Poin'
12494:   +'ter; ExceptObject : TObject; ExceptionRecord : PExceptionRecord; end
12495:   Procedure SafeCloseHandle( var Handle : THandle)
12496:   Procedure ExchangeInteger( X1, X2 : Integer)
12497:   Procedure FillInteger( const Buffer, Size, Value : Integer)
12498:   Function LongMulDiv( Mult1, Mult2, Div1 : Longint) : Longint
12499:   Function afCompareMem( P1, P2 : TObject; Length : Integer) : Boolean
12500:
12501: FILENAME_ADVAPI32 = 'ADVAPI32.DLL';
12502:   function AbortSystemShutdown; external advapi32 name 'AbortSystemShutdownW';
12503:   function AbortSystemShutdown(lpMachineName: PKOLChar): BOOL; stdcall;
12504:   function AccessCheckAndAuditAlarm(SubsystemName: PKOLChar;
12505:   HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12506:   SecurityDescriptor: PSecurityDescriptor; DesiredAccess: DWORD;
12507:   const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12508:   var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12509:   function AccessCheckByTypeAndAuditAlarm(SubsystemName: PKOLChar;
12510:   HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12511:   SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12512:   AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12513:   ObjectTypeNameLength: DWORD; const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12514:   var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12515:   function AccessCheckByTypeResultListAndAuditAlarm(SubsystemName: PKOLChar;
12516:   HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12517:   SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12518:   AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12519:   ObjectTypeNameLength: DWORD; const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12520:   var GrantedAccess: DWORD; var AccessStatusList: DWORD; var pfGenerateOnClose: BOOL): BOOL; stdcall;
12521:   function BackupEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12522:   function ClearEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12523:   function CreateProcessAsUser(hToken: THandle; lpApplicationName: PKOLChar;
12524:   lpCommandLine: PKOLChar; lpProcessAttributes: PSecurityAttributes;
12525:   lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL;
12526:   dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PKOLChar;
12527:   const lpStartupInfo: TStartupInfo; var lpProcessInformation: TProcessInformation): BOOL; stdcall;
12528:   function GetCurrentHwProfile(var lpHwProfileInfo: THWPProfileInfo): BOOL; stdcall;
12529:   function GetFileSecurity(lpFileName: PKOLChar; RequestedInformation: SECURITY_INFORMATION;
12530:   pSecurityDescriptor: PSecurityDescriptor; nLength: DWORD; var lpnLengthNeeded: DWORD): BOOL; stdcall;
12531:   function GetUserName(lpBuffer: PKOLChar; var nSize: DWORD): BOOL; stdcall;
12532:   function InitiateSystemShutdown(lpMachineName, lpMessage: PKOLChar;
12533:   dwTimeout: DWORD; bForceAppsClosed, bRebootAfterShutdown: BOOL): BOOL; stdcall;
12534:   function LogonUser(lpszUsername, lpszDomain, lpszPassword: PKOLChar;
12535:   dwLogonType, dwLogonProvider: DWORD; var phToken: THandle): BOOL; stdcall;
12536:   function LookupAccountName(lpSystemName, lpAccountName: PKOLChar;
12537:   Sid: PSID; var cbSid: DWORD; ReferencedDomainName: PKOLChar;
12538:   var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12539:   function LookupAccountSid(lpSystemName: PKOLChar; Sid: PSID;
12540:   Name: PKOLChar; var cbName: DWORD; ReferencedDomainName: PKOLChar;
12541:   var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12542:   function LookupPrivilegeDisplayName(lpSystemName, lpName: PKOLChar;
12543:   lpDisplayName: PKOLChar; var cbDisplayName, lpLanguageId: DWORD): BOOL; stdcall;
12544:   function LookupPrivilegeName(lpSystemName: PKOLChar;
12545:   var lpLuid: TLargeInteger; lpName: PKOLChar; var cbName: DWORD): BOOL; stdcall;
12546:   function LookupPrivilegeValue(lpSystemName, lpName: PKOLChar;
12547:   var lpLuid: TLargeInteger): BOOL; stdcall;
12548:   function ObjectCloseAuditAlarm(SubsystemName: PKOLChar;
12549:   HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12550:   function ObjectDeleteAuditAlarm(SubsystemName: PKOLChar;
12551:   HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12552:   function ObjectOpenAuditAlarm(SubsystemName: PKOLChar; HandleId: Pointer;
12553:   ObjectTypeName: PKOLChar; ObjectName: PKOLChar; pSecurityDescriptor: PSecurityDescriptor;
12554:   ClientToken: THandle; DesiredAccess, GrantedAccess: DWORD;
12555:   var Privileges: TPrivilegeSet; ObjectCreation, AccessGranted: BOOL;
12556:   var GenerateOnClose: BOOL): BOOL; stdcall;
12557:   function ObjectPrivilegeAuditAlarm(SubsystemName: PKOLChar;
12558:   HandleId: Pointer; ClientToken: THandle; DesiredAccess: DWORD;
12559:   var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;

```

```

12560: function OpenBackupEventLog(lpUNCServerName, lpFileName: PKOLChar): THandle; stdcall;
12561: function OpenEventLog(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12562: function PrivilegedServiceAuditAlarm(SubsystemName, ServiceName: PKOLChar;
12563: ClientToken: THandle; var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12564: function ReadEventLog(hEventLog: THandle; dwReadFlags, dwRecordOffset: DWORD;
12565: lpBuffer: Pointer; nNumberOfBytesToRead: DWORD;
12566: var pnBytesRead, pnMinNumberOfBytesNeeded: DWORD): BOOL; stdcall;
12567: function RegConnectRegistry(lpMachineName: PKOLChar; hKey: HKEY;
12568: var phkResult: HKEY): Longint; stdcall;
12569: function RegCreateKey(hKey: HKEY; lpSubKey: PKOLChar;
12570: var phkResult: HKEY): Longint; stdcall;
12571: function RegCreateKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12572: Reserved: DWORD; lpClass: PKOLChar; dwOptions: DWORD; samDesired: REGSAM;
12573: lpSecurityAttributes: PSecurityAttributes; var phkResult: HKEY;
12574: lpDisposition: PDWORD): Longint; stdcall;
12575: function RegDeleteKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12576: function RegDeleteValue(hKey: HKEY; lpValueName: PKOLChar): Longint; stdcall;
12577: function RegEnumKeyEx(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar;
12578: var lpcbName: DWORD; lpReserved: Pointer; lpClass: PKOLChar;
12579: lpcbClass: PDWORD; lpftLastWriteTime: PFileTime): Longint; stdcall;
12580: function RegEnumKey(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar; cbName: DWORD): Longint; stdcall;
12581: function RegEnumValue(hKey: HKEY; dwIndex: DWORD; lpValueName: PKOLChar;
12582: var lpcbValueName: DWORD; lpReserved: Pointer; lpType: PDWORD;
12583: lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12584: function RegLoadKey(hKey: HKEY; lpSubKey, lpFile: PKOLChar): Longint; stdcall;
12585: function RegOpenKey(hKey: HKEY; lpSubKey: PKOLChar; var phkResult: HKEY): Longint; stdcall;
12586: function RegOpenKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12587: ulOptions: DWORD; samDesired: REGSAM; var phkResult: HKEY): Longint; stdcall;
12588: function RegQueryInfoKey(hKey: HKEY; lpClass: PKOLChar;
12589: lpcbClass: PDWORD; lpReserved: Pointer;
12590: lpSubKeys, lpcbMaxSubKeyLen, lpcbMaxClassLen, lpValues,
12591: lpcbMaxValueNameLen, lpcbMaxValueLen, lpSecurityDescriptor: PDWORD;
12592: lpftLastWriteTime: PFileTime): Longint; stdcall;
12593: function RegQueryMultipleValues(hKey: HKEY; var ValList;
12594: NumVals: DWORD; lpValueBuf: PKOLChar; var ldwTotSize: DWORD): Longint; stdcall;
12595: function RegQueryValue(hKey: HKEY; lpSubKey: PKOLChar;
12596: lpValue: PKOLChar; var lpcbValue: Longint): Longint; stdcall;
12597: function RegQueryValueEx(hKey: HKEY; lpValueName: PKOLChar;
12598: lpReserved: Pointer; lpType: PDWORD; lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12599: function RegReplaceKey(hKey: HKEY; lpSubKey: PKOLChar;
12600: lpNewFile: PKOLChar; lpOldFile: PKOLChar): Longint; stdcall;
12601: function RegRestoreKey(hKey: HKEY; lpFile: PKOLChar; dwFlags: DWORD): Longint; stdcall;
12602: function RegSaveKey(hKey: HKEY; lpFile: PKOLChar;
12603: lpSecurityAttributes: PSecurityAttributes): Longint; stdcall;
12604: function RegSetValue(hKey: HKEY; lpSubKey: PKOLChar;
12605: dwType: DWORD; lpData: PKOLChar; cbData: DWORD): Longint; stdcall;
12606: function RegSetValueEx(hKey: HKEY; lpValueName: PKOLChar;
12607: Reserved: DWORD; dwType: DWORD; lpData: Pointer; cbData: DWORD): Longint; stdcall;
12608: function RegUnLoadKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12609: function RegisterEventSource(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12610: function ReportEvent(hEventLog: THandle; wType, wCategory: Word;
12611: dwEventID: DWORD; lpUserSid: Pointer; wNumStrings: Word;
12612: dwDataSize: DWORD; lpStrings, lpRawData: Pointer): BOOL; stdcall;
12613: function SetFileSecurity(lpFileName: PKOLChar; SecurityInformation: SECURITY_INFORMATION;
12614: pSecurityDescriptor: PSecurityDescriptor): BOOL; stdcall;
12615:
12616: Function wAddAtom( lpString : PKOLChar ) : ATOM
12617: Function wBeginUpdateResource( pFileName : PKOLChar; bDeleteExistingResources : BOOL ) : THandle
12618: //Function wCallNamedPipe( lpNamedPipeName : PKOLChar; lpInBuffer : Pointer; nInBufferSize : DWORD;
12619: lpOutBuffer : Pointer; nOutBufferSize : DWORD; var lpBytesRead : DWORD; nTimeout : DWORD ) : BOOL
12620: //Function wCommConfigDialog( lpszName : PKOLChar; hWnd : HWND; var lpCC : TCommConfig ) : BOOL
12621: Function wCompareString( Locale : LCID; dwCmpFlags : DWORD; lpString1 : PKOLChar; cchCount1 : Integer;
12622: lpString2 : PKOLChar; cchCount2 : Integer ) : Integer
12623: Function wCopyFile( lpExistingFileName, lpNewFileName : PKOLChar; bFailIfExists : BOOL ) : BOOL
12624: //Function wCopyFileEx( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
12625: TFNProgressRoutine; lpData : Pointer; pbCancel : PBool; dwCopyFlags : DWORD ) : BOOL
12626: Function wCreateDirectory( lpPathName : PKOLChar; lpSecurityAttributes : PSecurityAttributes ) : BOOL
12627: Function wCreateDirectoryEx( lpTemplateDirectory,
12628: lpNewDirectory:PKOLChar;lpSecAttrib:PSecurityAttribs):BOOL;
12629: Function wCreateEvent(lpEventAttribs:PSecurityAttrib;bManualReset,
12630: bInitialState:BOOL;lpName:PKOLChar):THandle;
12631: Function wCreateFile( lpFileName : PKOLChar; dwDesiredAccess, dwShareMode : DWORD; lpSecurityAttributes :
12632: PSecurityAttributes; dwCreationDisposition, dwFlagsAndAttributes:DWORD;hTemplateFile:THandle):THandle
12633: Function wCreateFileMapping( hFile : THandle; lpFileMappingAttributes : PSecurityAttributes; flProtect,
12634: dwMaximumSizeHigh, dwMaximumSizeLow : DWORD; lpName : PKOLChar ) : THandle
12635: Function wCreateHardLink(lpFileName,
12636: lpExistingFileName:PKOLChar;lpSecurityAttributes:PSecurityAttributes):BOOL
12637: Function
12638: CreateMailslot(lpName:PKOLChar;MaxMessSize:DWORD;lReadTimeout:DWORD;lpSecurityAttrib:PSecurityAttributes):THandle);
12639: Function wCreateNamedPipe( lpName : PKOLChar; dwOpenMode, dwPipeMode, nMaxInstances, nOutBufferSize,
12640: nInBufferSize, nDefaultTimeOut : DWORD; lpSecurityAttributes : PSecurityAttributes ) : THandle
12641: //Function CreateProcess( lpApplicationName : PKOLChar; lpCommandLine : PKOLChar; lpProcessAttributes,
12642: lpThreadAttributes : PSecurityAttributes; bInheritHandles : BOOL; dwCreationFlags : DWORD; lpEnvironment :
12643: Pointer; lpCurrentDirectory:PKOLChar;const lpStartupInfo:TStartupInfo;var
12644: lpProcessInfo:TProcessInformation): BOOL
12645: Function wCreateSemaphore(lpSemaphoreAttributes: PSecurityAttributes; lInitialCount, lMaximumCount :
12646: Longint; lpName : PKOLChar ) : THandle
12647: Function
12648: wCreateWaitableTimer(lpTimerAttributes:PSecurityAttribs;bManualReset:BOOL;lpTimerName:PKOLChar):THandle);

```

```

12634: Function wDefineDosDevice( dwFlags : DWORD; lpDeviceName, lpTargetPath : PKOLChar ) : BOOL
12635: Function wDeleteFile( lpFileName : PKOLChar ) : BOOL
12636: Function wEndUpdateResource( hUpdate : THandle; fDiscard : BOOL ) : BOOL
12637: //Function
wEnumCalendarInfo( lpCalInfEnumProc: TFNCalInfEnumProc; Locale: LCID; Calendar: CALID; CalType: CALTYPE ): BOOL;
12638: //Function wEnumDateFormats( lpDateFmtEnumProc: TFNDateFmtEnumProc; Locale : LCID; dwFlags : DWORD ) : BOOL
12639: // Function wEnumResourceLanguages( hModule: HMODULE; lpType,
lpName: PChar; lpEnumFunc: ENUMRESLANGPROC; lParam: Longint: BOOL' )
12640: //Function
wEnumResourceNames( hModule: HMODULE; lpType: PKOLChar; lpEnumFunc: ENUMRESNAMEPROC; lParam: Longint ): BOOL;
12641: //Function wEnumResourceTypes( hModule: HMODULE; lpEnumFunc: ENUMRESTYPEPROC; lParam: Longint ): BOOL;
12642: //Function wEnumSystemCodePages( lpCodePageEnumProc : TFNCODEPAGEENUMPROC; dwFlags : DWORD ) : BOOL
12643: //Function wEnumSystemLocales( lpLocaleEnumProc : TFNLOCALEENUMPROC; dwFlags : DWORD ) : BOOL
12644: //Function wEnumTimeFormats( lpTimeFmtEnumProc: TFNTIMEFMTENUMPROC; Locale: LCID; dwFlags: DWORD ): BOOL;
12645: Function wExpandEnvironmentStrings( lpSrc : PKOLChar; lpDst : PKOLChar; nSize : DWORD ) : DWORD
12646: Procedure wFatalAppExit( uAction : UINT; lpMessageText : PKOLChar)
12647: //Function wFillConsoleOutputCharacter( hConsoleOutput : THandle; cCharacter : KOLChar; nLength : DWORD;
dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD ) : BOOL
12648: Function wFindAtom( lpString : PKOLChar ) : ATOM
12649: Function
wFindFirstChangeNotification( lpPathName: PKOLChar; bWatchSubtree: BOOL; dwNotifyFilter: DWORD ): THandle;
12650: Function wFindFirstFile( lpFileName : PKOLChar; var lpFindFileData : TWIN32FindData ) : THandle
12651: //Function wFindFirstFileEx( lpFileName : PKOLChar; fInfoLevelId : TFindIndexInfoLevels; lpFindFileData :
Pointer; fSearchOp : TFindIndexSearchOps; lpSearchFilter : Pointer; dwAdditionalFlags : DWORD ) : BOOL
12652: Function wFindNextFile( hFindFile : THandle; var lpFindFileData : TWIN32FindData ) : BOOL
12653: Function wFindResource( hModule : HMODULE; lpName, lpType : PKOLChar ) : HRSRC
12654: Function wFindResourceEx( hModule : HMODULE; lpType, lpName : PKOLChar; wLanguage : Word ) : HRSRC
12655: Function
wFoldString( dwMapFlags: DWORD; lpSrcStr: PKOLChar; cchSrc: Int; lpDestStr: PKOLChar; cchDest: Integer ): Integer;
12656: //Function wFormatMessage( dwFlags : DWORD; lpSource : Pointer; dwMessageId : DWORD; dwLanguageId :
DWORD; lpBuffer : PKOLChar; nSize : DWORD; Arguments : Pointer ) : DWORD
12657: Function wFreeEnvironmentStrings( EnvBlock : PKOLChar ) : BOOL
12658: Function wGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer ) : UINT
12659: Function wGetBinaryType( lpApplicationName : PKOLChar; var lpBinaryType : DWORD ) : BOOL
12660: Function wGetCommandLine : PKOLChar
12661: //Function wGetCompressedFileSize( lpFileName : PKOLChar; lpFileSizeHigh : PDWORD ) : DWORD
12662: Function wGetComputerName( lpBuffer : PKOLChar; var nSize : DWORD ) : BOOL
12663: Function wGetConsoleTitle( lpConsoleTitle : PKOLChar; nSize : DWORD ) : DWORD
12664: //Function wGetCurrencyFormat( Locale : LCID; dwFlags : DWORD; lpValue : PKOLChar; lpFormat :
PCurrencyFmt; lpCurrencyStr : PKOLChar; cchCurrency : Integer ) : Integer
12665: Function wGetCurrentDirectory( nBufferLength : DWORD; lpBuffer : PKOLChar ) : DWORD
12666: //Function wGetDateFormat( Locale : LCID; dwFlags : DWORD; lpDate : PSystemTime; lpFormat : PKOLChar;
lpDateStr : PKOLChar; cchDate : Integer ) : Integer
12667: //Function wGetDefaultCommConfig( lpSzName: PKOLChar; var lpCC : TCommConfig; var lpDwSize: DWORD ): BOOL
12668: Function wGetDiskFreeSpace( lpRootPathName : PKOLChar; var lpSectorsPerCluster, lpBytesPerSector,
lpNumberOfFreeClusters, lpTotalNumberOfClusters : DWORD ) : BOOL
12669: //Function wGetDiskFreeSpaceEx( lpDirectoryName : PKOLChar; var lpFreeBytesAvailableToCaller,
lpTotalNumberOfBytes, lpTotalNumberOfFreeBytes : PLargeInteger ) : BOOL
12670: Function wGetDriveType( lpRootPathName : PKOLChar ) : UINT
12671: Function wGetEnvironmentStrings : PKOLChar
12672: Function wGetEnvironmentVariable( lpName : PKOLChar; lpBuffer : PKOLChar; nSize : DWORD ) : DWORD;
12673: Function wGetFileAttributes( lpFileName : PKOLChar ) : DWORD
12674: //Function
wGetFileAttributesEx( lpFileName: PKOLChar; fInfoLevelId: TGetFileExInfoLevs; lpFileInform: Pointer ): BOOL;
12675: Function wGetFullPathName( lpFileName: PKOLChar; nBufferLength: WORD; lpBuffer: PKOLChar; var
lpFilePart: PKOLChar ): DWORD;
12676: //Function wGetLocaleInfo( Locale: LCID; LCTYPE: LCTYPE; lpLCData: PKOLChar; cchData: Integer ): Integer
12677: Function wGetLogicalDriveStrings( nBufferLength : DWORD; lpBuffer : PKOLChar ) : DWORD
12678: Function wGetModuleFileName( hModule : HINST; lpFilename : PKOLChar; nSize : DWORD ) : DWORD
12679: Function wGetModuleHandle( lpModuleName : PKOLChar ) : HMODULE
12680: //Function wGetNamedPipeHandleState( hNamedPipe : THandle; lpState, lpCurInstances, lpMaxCollectionCount,
lpCollectDataTimeout : PDWORD; lpUserName : PKOLChar; nMaxUserNameSize : DWORD ) : BOOL
12681: //Function wGetNumberFormat( Locale : LCID; dwFlags: DWORD; lpValue: PKOLChar; lpFormat: PNumberFmt;
lpNumberStr : PKOLChar; cchNumber : Integer ) : Integer
12682: Function wGetPrivateProfileInt( lpAppName, lpKeyName: PKOLChar; nDefault: Integer; lpFileName: PKOLChar ): UINT;
12683: Function
wGetPrivateProfileSection( lpAppName: PKOLChar; lpRetrStr: PKOLChar; nSize: DWORD; pFileName: PKOLChar ): DWORD;
12684: Function wGetPrivateProfileSectionNames( lpSzReturnBuffer: PKOLChar; nSize: DWORD; lpFileName: PKOLChar ): DWORD;
12685: Function wGetPrivateProfileString( lpAppName, lpKeyName, lpDefault : PKOLChar; lpReturnedStr : PKOLChar;
nSize: DWORD; lpFileName : PKOLChar ) : DWORD
12686: Function wGetProfileInt( lpAppName, lpKeyName : PKOLChar; nDefault : Integer ) : UINT
12687: Function wGetProfileSection( lpAppName : PKOLChar; lpReturnedString : PKOLChar; nSize : DWORD ) : DWORD
12688: Function wGetProfileString( lpAppName, lpKeyName,
lpDefault: PKOLChar; lpReturnedStr: PKOLChar; nSize: DWORD ): DWORD;
12689: Function wGetShortPathName( lpSzLongPath: PKOLChar; lpSzShortPath : PKOLChar; cchBuffer : DWORD ) : DWORD
12690: //Procedure wGetStartupInfo( var lpStartupInfo : TStartupInfo)
12691: // Function wGetStringTypeEx( Locale: LCID; dwInfoType: DWORD; lpSrcStr: PKOLChar; cchSrc: Integer; var
lpCharType ): BOOL
12692: Function wGetSystemDirectory( lpBuffer : PKOLChar; uSize : UINT ) : UINT
12693: Function wGetTempFileName( lpPathName, lpPrefixString: PKOLChar; uUnique: UINT; lpTempFileName: PKOLChar ): UINT
12694: Function wGetTempPath( nBufferLength : DWORD; lpBuffer : PKOLChar ) : DWORD
12695: //Function
wGetTimeFormat( Loc: LCID; dwFlgs: DWORD; lpTime: PSystemTime; lpFrm: PKOLChar; lpTimeStr: PKOLChar; cTime: Int ): Int
12696: //Function wGetVersionEx( var lpVersionInformation : TOSVersionInfo ) : BOOL
12697: //Function wGetVolumeInformation( lpRootPathName : PKOLChar; lpVolumeNameBuffer : PKOLChar; nVolumeNameSize :
DWORD; lpVolumeSerialNumber : PDWORD; var lpMaximumComponentLength, lpFileSystemFlags : DWORD;
lpFileSystemNameBuffer : PKOLChar; nFileSystemNameSize : DWORD ) : BOOL
12698: Function wGetWindowsDirectory( lpBuffer : PKOLChar; uSize : UINT ) : UINT
12699: Function wGlobalAddAtom( lpString : PKOLChar ) : ATOM

```



```

12700: Function wGlobalFindAtom( lpString : PKOLChar ) : ATOM
12701: Function wGlobalGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer ) : UINT
12702: Function wIsBadStringPtr( lpsz : PKOLChar; ucchMax : UINT ) : BOOL
12703: Function
wLcMapString( Loc:LCID;dwMapFlgs:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Int):Int;
12704: Function wLoadLibrary( lpLibFileName : PKOLChar ) : HMODULE
12705: Function wLoadLibraryEx( lpLibFileName : PKOLChar; hFile : THandle; dwFlags : DWORD ) : HMODULE
12706: Function wMoveFile( lpExistingFileName, lpNewFileName : PKOLChar ) : BOOL
12707: Function wMoveFileEx( lpExistingFileName, lpNewFileName : PKOLChar; dwFlags : DWORD ) : BOOL
12708: //Function wMoveFileWithProgress( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
TFNProgressRoutine; lpData : Pointer; dwFlags : DWORD ) : BOOL
12709: Function wOpenEvent( dwDesiredAccess : DWORD; binheritHandle : BOOL; lpName:PKOLChar ) : THandle
12710: Function wOpenFileMapping( dwDesiredAccess : DWORD; binheritHandle:BOOL;lpName: PKOLChar):THandle
12711: Function wOpenMutex( dwDesiredAccess : DWORD; binheritHandle : BOOL; lpName : PKOLChar ) : THandle
12712: Function wOpenSemaphore( dwDesiredAccess : DWORD; binheritHandle : BOOL; lpName : PKOLChar):THandle
12713: Function wOpenWaitableTimer(dwDesiredAccess:DWORD;binheritHandle:BOOL;lpTimerName:PKOLChar):THandle
12714: Procedure wOutputDebugString( lpOutputString : PKOLChar)
12715: //Function wPeekConsoleInput(hConsoleInput:THandle;var lpBuffer:TInputRecord;nLength:DWORD;var
lpNumberOfEventsRead:DWORD):BOOL;
12716: Function wQueryDosDevice( lpDeviceName : PKOLChar; lpTargetPath : PKOLChar; ucchMax : DWORD ) : DWORD
12717: //Function wQueryRecoveryAgents(p1:PKOLChar;var p2:Pointer;var p3:TRecoveryAgentInformation):DWORD
12718: //Function wReadConsole( hConsoleInput : THandle; lpBuffer : Pointer; nNumberOfCharsToRead : DWORD; var
lpNumberOfCharsRead : DWORD; lpReserved : Pointer ) : BOOL
12719: //Function wReadConsoleInput(hConsInp:THandle;var lpBuf:TInpRec;nLength:DWORD;var
lpNumberOfEventsRead:DWORD):BOOL;
12720: //Function wReadConsoleOutput( hConsoleOutput : THandle; lpBuffer : Pointer; dwBufferSize, dwBufferCoord
: TCoord; var lpReadRegion : TSmallRect ) : BOOL
12721: //Function wReadConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
DWORD; dwReadCoord : TCoord; var lpNumberOfCharsRead : DWORD ) : BOOL
12722: Function wRemoveDirectory( lpPathName : PKOLChar ) : BOOL
12723: //Function wScrollConsoleScreenBuffer( hConsoleOutput : THandle; const lpScrollRectangle : TSmallRect;
lpClipRectangle : PSmallRect; dwDestinationOrigin : TCoord; var lpFill : TCharInfo ) : BOOL
12724: Function wSearchPath( lpPath, lpFileName, lpExtension:PKOLChar;nBufferLength:DWORD;lpBuffer:PKOLChar;var
lpFilePart:PKOLChar):DWORD;
12725: Function wSetComputerName( lpComputerName : PKOLChar ) : BOOL
12726: Function wSetConsoleTitle( lpConsoleTitle : PKOLChar ) : BOOL
12727: Function wSetCurrentDirectory( lpPathName : PKOLChar ) : BOOL
12728: //Function wSetDefaultCommConfig( lpszName : PKOLChar; lpCC : PCommConfig; dwSize : DWORD ) : BOOL
12729: Function wSetEnvironmentVariable( lpName, lpValue : PKOLChar ) : BOOL
12730: Function wSetFileAttributes( lpFileName : PKOLChar; dwFileAttributes : DWORD ) : BOOL
12731: //Function wSetLocaleInfo( Locale : LCID; LCTYPE : LCTYPE; lpLCData : PKOLChar ) : BOOL
12732: Function wSetVolumeLabel( lpRootPathName : PKOLChar; lpVolumeName : PKOLChar ) : BOOL
12733: //Function wUpdateResource(hUpdate:THandle;lpType,
lpName:PKOLChar;wLanguage:Word;lpData:Ptr;cbData:DWORD):BOOL
12734: Function wVerLanguageName( wLang : DWORD; szLang : PKOLChar; nSize : DWORD ) : DWORD
12735: Function wWaitNamedPipe( lpNamedPipeName : PKOLChar; nTimeout : DWORD ) : BOOL
12736: //Function wWriteConsole( hConsoleOutput : THandle; const lpBuffer : Pointer; nNumberOfCharsToWrite :
DWORD; var lpNumberOfCharsWritten : DWORD; lpReserved : Pointer ) : BOOL
12737: //Function wWriteConsoleInput( hConsoleInput : THandle; const lpBuffer : TInputRecord; nLength : DWORD;
var lpNumberOfEventsWritten : DWORD ) : BOOL
12738: //Function wWriteConsoleOutput(hConsoleOutput:THandle; lpBuffer:Pointer; dwBufferSize,dwBufferCoord :
TCoord; var lpWriteRegion : TSmallRect ) : BOOL
12739: //Function wWriteConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
DWORD; dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD ) : BOOL
12740: Function wWritePrivateProfileSection( lpAppName, lpString, lpFileName : PKOLChar ) : BOOL
12741: Function wWritePrivateProfileString( lpAppName, lpKeyName, lpString, lpFileName : PKOLChar ) : BOOL
12742: Function wWriteProfileSection( lpAppName, lpString : PKOLChar ) : BOOL
12743: Function wWriteProfileString( lpAppName, lpKeyName, lpString : PKOLChar ) : BOOL
12744: Function wlstreat( lpString1, lpString2 : PKOLChar ) : PKOLChar
12745: Function wlstrcmp( lpString1, lpString2 : PKOLChar ) : Integer
12746: Function wlstrcmpi( lpString1, lpString2 : PKOLChar ) : Integer
12747: Function wlstropy( lpString1, lpString2 : PKOLChar ) : PKOLChar
12748: Function wlstropyn( lpString1, lpString2 : PKOLChar; iMaxLength : Integer ) : PKOLChar
12749: Function wlstrlen( lpString : PKOLChar ) : Integer
12750: //Function wMultinetGetConnectionPerformance( lpNetResource : PNetResource; lpNetConnectInfoStruc :
PNetConnectInfoStruct ) : DWORD
12751: //Function wWNetAddConnection2(var lpNetResource:TNetResource;lpPassword,
lpUserName:PKOLChar;dwFlags:DWORD):DWORD;
12752: //Function wWNetAddConnection3( hwndOwner : HWND; var lpNetResource:TNetResource;lpPassword,
lpUserName:PKOLChar; dwFlags : DWORD ) : DWORD
12753: Function wWNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PKOLChar ) : DWORD
12754: Function wWNetCancelConnection2( lpName : PKOLChar; dwFlags : DWORD; fForce : BOOL ) : DWORD
12755: Function wWNetCancelConnection( lpName : PKOLChar; fForce : BOOL ) : DWORD
12756: //Function wWNetConnectionDialog1( var lpConnDlgStruct : TConnectDlgStruct ) : DWORD
12757: //Function wWNetDisconnectDialog1( var lpConnDlgStruct : TDiscDlgStruct ) : DWORD
12758: //Function wWNetEnumResource(hEnum:THandle;var lpCount:DWORD;lpBuffer:Ptr;var lpBufferSize:DWORD):DWORD;
12759: Function wWNetGetConnection(lpLocalName:PKOLChar;lpRemoteName:PKOLChar; var lpnLength:DWORD):DWORD;
12760: Function wWNetGetLastError( var lpError : DWORD; lpErrorBuf : PKOLChar; nErrorBufSize : DWORD; lpNameBuf
: PKOLChar; nNameBufSize : DWORD ) : DWORD
12761: //Function wWNetGetNetworkInformation(lpProvider:PKOLChar;var lpNetInfoStruct:TNetInfoStruct):DWORD;
12762: Function wWNetGetProviderName(dwNetType:DWORD;lpProviderName:PKOLChar;var lpBufferSize:DWORD):DWORD;
12763: //Function wWNetGetResourceParent(lpNetResource:PNetResource;lpBuffer:Pointer;var cbBuffer:DWORD):DWORD;
12764: //Function wWNetGetUniversalName(lpLocalPath:PKOLChar;dwInfoLevel:DWORD;lpBuffer:Ptr;var
lpBufferSize:DWORD):DWORD;
12765: Function wWNetGetUser( lpName : PKOLChar; lpUserName : PKOLChar; var lpnLength : DWORD ) : DWORD
12766: //Function wWNetOpenEnum(dwScope,dwType,dwUsage:DWORD;lpNetResource:PNetRes;var lphEnum:THandle):DWORD;
12767: //Function wWNetSetConnection( lpName : PKOLChar; dwProperties : DWORD; pvValues : Pointer ) : DWORD
12768: //Function wWNetUseConnection(hwndOwner:HWND;var
lpNetResource:TNetResource;lpUserID:PKOLChar;lpPassword:PKOLChar; dwFlags:DWORD;lpAccessName:PKOLChar;var
lpBufferSize:DWORD;var lpResult:DWORD):DWORD

```



```

12769: Function wGetFileVersionInfo(lptstrFilename:PKOLChar;dwHandle,dwLen:DWORD;lpData:Pointer):BOOL;
12770: Function wGetFileVersionInfoSize( lptstrFilename : PKOLChar; var lpdwHandle : DWORD ) : DWORD
12771: Function wVerFindFile( uFlags : DWORD; szFileName, szWinDir, szAppDir, szCurDir : PKOLChar; var
lpuCurDirLen : UINT; szDestDir : PKOLChar; var lpDestDirLen : UINT ) : DWORD
12772: Function wVerInstallFile( uFlags : DWORD; szSrcFileName, szDestFileName, szSrcDir, szDestDir, szCurDir,
szTmpFile : PKOLChar; var lpuTmpFileLen : UINT ) : DWORD
12773: //Function wVerQueryValue(pBlock:Pointer;lpSubBlock:PKOLChar;var lpBuffer:Ptr;var puLen:UINT):BOOL;
12774: //Function wGetPrivateProfileStruct(lpszSection,
lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12775: //Function wWritePrivateProfileStruct(lpszSection,
lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12776: Function wAddFontResource( FileName : PKOLChar ) : Integer
12777: //Function wAddFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector ) : Integer
12778: Function wCopyEnhMetaFile( p1 : HENHMETAFILE; p2 : PKOLChar ) : HENHMETAFILE
12779: Function wCopyMetaFile( p1 : HMETAFILE; p2 : PKOLChar ) : HMETAFILE
12780: //Function wCreateColorSpace( var ColorSpace : TLogColorSpace ) : HCOLORSPACE
12781: //Function wCreateDC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvminit : PDeviceMode ) : HDC
12782: // Function wCreateEnhMetaFile( DC : HDC; FileName : PKOLChar; Rect : PRect; Desc : PKOLChar ) : HDC
12783: Function wCreateFont( nHeight, nWidth, nEscapement, nOrientaion, fnWeight : Integer; fdwItalic,
fdwUnderline, fdwStrikeOut, fdwCharSet, fdwOutputPrec, fdwClipPrecision, fdwQualy,
fdwPitchAndFamily:DWORD;lpszFace:PKOLChar):HFONT;
12784: Function wCreateFontIndirect( const p1 : TLogFont ) : HFONT
12785: //Function wCreateFontIndirectEx( const p1 : PEnumLogFontExDV ) : HFONT
12786: // Function wCreateIC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvminit : PDeviceMode ) : HDC
12787: Function wCreateMetaFile( p1 : PKOLChar ) : HDC
12788: Function wCreateScalableFontResource( p1 : DWORD; p2, p3, p4 : PKOLChar ) : BOOL
12789: //Function wDeviceCapabilities(pDriverNa,pDeviceNam,
pPort:PKOLChar;iIdx:Int;pOut:PKOLChar;DevMod:PDeviceMode):Int;
12790: // Function wEnumFontFamilies( DC : HDC; p2 : PKOLChar; p3 : TFNFontEnumProc; p4 : LPARAM ) : BOOL
12791: //Function wEnumFontFamiliesEx(DC:HDC;var p2:TLogFont;p3:TFNFontEnumProc;p4:LPARAM;p5:DWORD):BOOL);
12792: //Function wEnumFonts(DC:HDC;lpszFace:PKOLChar;Entenmpcr:TFNFontEnumProc;lpszData:PKOLChar):Integer;
12793: //Function wEnumICMPProfiles( DC : HDC; ICMPProc : TFNICMEnumProc; p3 : LPARAM ) : Integer
12794: //Function wExtTextOut(DC:HDC;X,
Y:Int;Options:Longint;Rect:PRect;Str:PKOLChar;Count:Longint;Dx:PInteger:BOOL
12795: //Function wGetCharABCWidths( DC : HDC; FirstChar, LastChar : UINT; const ABCStructs ) : BOOL
12796: //Function wGetCharABCWidthsFloat( DC : HDC; FirstChar, LastChar : UINT; const ABCFloatSturcts ) : BOOL
12797: //Function wGetCharWidth32( DC : HDC; FirstChar, LastChar : UINT; const Widths ) : BOOL
12798: //Function wGetCharWidth( DC : HDC; FirstChar, LastChar : UINT; const Widths ) : BOOL
12799: // Function wGetCharWidthFloat( DC : HDC; FirstChar, LastChar : UINT; const Widths ) : BOOL
12800: // Function wGetCharacterPlacement(DC:HDC;p2:PKOLChar;p3,p4:BOOL;var p5:TGCPRResults;p6:DWORD):DWORD
12801: Function wGetEnhMetaFile( p1 : PKOLChar ) : HENHMETAFILE
12802: Function wGetEnhMetaFileDescription( p1 : HENHMETAFILE; p2 : UINT; p3 : PKOLChar ) : UINT
12803: // Function wGetGlyphIndices( DC : HDC; p2 : PKOLChar; p3 : Integer; p4 : PWORD; p5 : DWORD ) : DWORD
12804: // Function wGetGlyphOutline( DC : HDC; uChar,uFormat:UINT;const lpgm:TGlyphMetrics; cbBuffer : DWORD;
lpvBuffer : Pointer; const lpmat2 : TMat2 ) : DWORD
12805: Function wGetICMPProfile( DC : HDC; var Size : DWORD; Name : PKOLChar ) : BOOL
12806: // Function wGetLogColorSpace( p1 : HCOLORSPACE; var ColorSpace : TLogColorSpace; Size : DWORD ) : BOOL
12807: Function wGetMetaFile( p1 : PKOLChar ) : HMETAFILE
12808: // Function wGetObject( p1 : HGDIOBJ; p2 : Integer; p3 : Pointer ) : Integer
12809: //Function wGetOutlineTextMetrics( DC : HDC; p2 : UINT; OTMetricStructs : Pointer ) : UINT
12810: //Function wGetTextExtentExPoint(DC:HDC;p2:PKOLChar; p3,p4:Integer;p5,p6:PInteger;var p7:TSize):BOOL
12811: Function wGetTextExtentPoint32( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize ) : BOOL
12812: Function wGetTextExtentPoint( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize ) : BOOL
12813: Function wGetTextFace( DC : HDC; Count : Integer; Buffer : PKOLChar ) : Integer
12814: //Function wGetTextMetrics( DC : HDC; var TM : TTextMetric ) : BOOL
12815: Function wPolyTextOut( DC : HDC; const PolyTextArray, Strings : Integer ) : BOOL
12816: Function wRemoveFontResource( FileName : PKOLChar ) : BOOL
12817: //Function wRemoveFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector ) : BOOL
12818: //Function wResetDC( DC : HDC; const InitData : TDeviceMode ) : HDC
12819: Function wSetICMPProfile( DC : HDC; Name : PKOLChar ) : BOOL
12820: //Function wStartDoc( DC : HDC; const p2 : TDocInfo ) : Integer
12821: Function wTextOut( DC : HDC; X, Y : Integer; Str : PKOLChar; Count : Integer ) : BOOL
12822: Function wUpdateICMRegKey( p1 : DWORD; p2, p3 : PKOLChar; p4 : UINT ) : BOOL
12823: Function wvglUseFontBitmaps( DC : HDC; p2, p3, p4 : DWORD ) : BOOL
12824: //Function wvglUseFontOutlines(p1:HDC; p2,p3,p4:DWORD;p5,p6:Single;p7:Int;p8:PGlyphMetricsFloat):BOOL
12825: Function wAppendMenu( hMenu : HMENU; uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar ) : BOOL
12826: Function wCallMsgFilter( var lpMsg : TMsg; nCode : Integer ) : BOOL
12827: //Function
wCallWindowProc( lpPrevWndFunc:TFNWndProc;hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT
12828: //Function wChangeDisplaySettings( var lpDevMode : TDeviceMode; dwFlags : DWORD ) : Longint
12829: // Function wChangeDisplaySettingsEx( lpszDeviceName:PKOLChar;var lpDevMode: TDeviceMode; wnd : HWND;
dwFlags : DWORD; lParam : Pointer ) : Longint
12830: Function wChangeMenu( hMenu:HMENU;cmd:UINT;lpszNewItem:PKOLChar;cmdInsert:UINT;flags:UINT):BOOL;
12831: Function wCharLower( lpsz : PKOLChar ) : PKOLChar
12832: Function wCharLowerBuff( lpsz : PKOLChar; cchLength : DWORD ) : DWORD
12833: Function wCharNext( lpsz : PKOLChar ) : PKOLChar
12834: //Function wCharNextEx( CodePage : Word; lpCurrentChar : LPCSTR; dwFlags : DWORD ) : LPSTR
12835: Function wCharPrev( lpszStart : PKOLChar; lpszCurrent : PKOLChar ) : PKOLChar
12836: // Function wCharPrevEx( CodePage : Word; lpStart, lpCurrentChar : LPCSTR; dwFlags : DWORD ) : LPSTR
12837: Function wCharToOem( lpszSrc : PKOLChar; lpszDst : PKOLChar ) : BOOL
12838: Function wCharToOemBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD ) : BOOL
12839: Function wCharUpper( lpsz : PKOLChar ) : PKOLChar
12840: Function wCharUpperBuff( lpsz : PKOLChar; cchLength : DWORD ) : DWORD
12841: Function wCopyAcceleratorTable( hAccelSrc : HACCEL; var lpAccelDst : Integer ) : Integer
12842: Function wCreateAcceleratorTable( var Accel, Count : Integer ) : HACCEL
12843: //Function wCreateDesktop(lpszDesktop,
lpszDevice:PKOLChar;pDevmode:PDeviceMode;dwFlags:DWORD;dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HDESK
12844: //Function wCreateDialogIndirectParam( hInstance : HINST; const lpTemplate : TDlgTemplate; hWndParent :
HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM ) : HWND

```

```

12845: //Function wCreateDialogParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND;
lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM ) : HWND
12846: Function wCreateMDIWindow( lpClassName, lpWindowName:PKOLChar;dwStyle: DWORD; X,Y,nWidth,nHeight:Integer;
hWndParent : HWND; hInstance : HINST; lParam : LPARAM ) : HWND
12847: //Function wCreateWindowEx(dwExStyle:DWORD;lpClassName:PKOLChar;lpWindowName:PKOLChar;dwStyle:DWORD;X,Y,
nWidth, nHeight:Int WndParent:HWND;hMenu:HMENU;hInstance:HINST;lpParam:Pointer):HWND
12848: //Function wCreateWindowStation(lpwinSta:PKOLChar;dwReserv,
dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HWINSTA;
12849: Function wDefDlgProc( hDlg : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT
12850: Function wDefFrameProc(hWnd,hWndMDIClient:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
12851: Function wDefMDIChildProc( hWnd : HWND; uMsg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT;
12852: Function wDefWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM): LRESULT
12853: //Function wDialogBoxIndirectParam( hInstance : HINST; const lpDialogTemplate : TDlgTemplate; hWndParent
: HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM ) : Integer
12854: //Function wDialogBoxParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND; lpDialogFunc
: TFNDlgProc; dwInitParam : LPARAM ) : Integer
12855: Function wDispatchMessage( const lpMsg : TMsg ) : Longint
12856: Function wDlgDirList(hDlg:HWND;lpPathSpec:PKOLChar;nIDListBox,
nIDStaticPath:Integer;uFileType:UINT):Integer;
12857: Function wDlgDirListComboBox(hDlg:HWND;lpPathSpec:PKOLChar;nIDComboBox,
nIDStaticPath:Int;uFileType:UINT):Int;
12858: Function wDlgDirSelectComboBoxEx( hDlg : HWND; lpString:PKOLChar;nCount,nIDComboBox:Integer): BOOL
12859: Function wDlgDirSelectEx( hDlg : HWND; lpString : PKOLChar; nCount, nIDListBox : Integer ) : BOOL
12860: //Function wDrawState(DC:HDC;Brush:HBRUSH;CBFunc:TFNDrawStateProc;lData:LPARA;wDat:WPARA;x,y,cx,
cy:Int;Flags:UINT):BOOL;
12861: Function wDrawText(hDC:HDC;lpString:PKOLChar;nCount:Integer;var lpRect:TRect;uFormat:UINT):Integer;
12862: Function wFindWindow( lpClassName, lpWindowName : PKOLChar ) : HWND
12863: Function wFindWindowEx( Parent, Child : HWND; ClassName, WindowName : PKOLChar ) : HWND
12864: //Function wGetAltTabInfo(hWnd:HWND;iItem:Int;var
pati:TAltTabInfo;pszItemText:PKOLChar;cchItemText:UINT):BOOL;
12865: //Function wGetClassInfo( hInstance : HINST; lpClassName : PKOLChar; var lpWndClass : TWndClass ) : BOOL
12866: //Function wGetClassInfoEx( Instance : HINST; Classname : PKOLChar; var WndClass : TWndClassEx ) : BOOL
12867: Function wGetClassLong( hWnd : HWND; nIndex : Integer ) : DWORD
12868: Function wGetClassName( hWnd : HWND; lpClassName : PKOLChar; nMaxCount : Integer ) : Integer
12869: Function wGetClipboardFormatName( format : UINT; lpszFormatName:PKOLChar;cchMaxCount:Integer):Integer;
12870: Function wGetDlgItemText( hDlg: HWND;nIDDlgItem:Integer;lpString:PKOLChar;nMaxCount:Integer):UINT
12871: Function wGetKeyNameText( lParam : Longint; lpString : PKOLChar; nSize : Integer ) : Integer
12872: Function wGetKeyboardLayoutName( pwszKLID : PKOLChar ) : BOOL
12873: //Function wGetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; var p4 : TMenuItemInfo ) : BOOL
12874: Function wGetMenuString(hMenu:HMENU;uIDItem:UINT;lpString:PKOLChar;nMaxCount:Integer;uFlag:UINT):Integer;
12875: Function wGetMessage( var lpMsg : TMsg; hWnd : HWND; wParam:FilterMin, wParam:FilterMax : UINT ) : BOOL
12876: Function wGetProp( hWnd : HWND; lpString : PKOLChar ) : THandle
12877: //Function wGetTabbedTextExtent( hDC : HDC; lpString : PKOLChar; nCount, nTabPositions : Integer; var
lpnTabStopPositions ) : DWORD
12878: //Function wGetObjectInformation(hObj:THandle;nIndex:Int;pvInfo:Ptr;nLength:DWORD;var
lpnLengthNeed:DWORD)BOOL;
12879: Function wGetWindowLong( hWnd : HWND; nIndex : Integer ) : Longint
12880: Function wGetWindowModuleFileName( hWnd : HWND; pszFileName : PKOLChar; cchFileNameMax : UINT ) : UINT
12881: Function wGetWindowText( hWnd : HWND; lpString : PKOLChar; nMaxCount : Integer ) : Integer
12882: Function wGetWindowTextLength( hWnd : HWND ) : Integer
12883: //Function wGrayString(hDC:HDC;hBrush:HBRUSH;lpOutFunc:TFNGrayStrProc;lpDat:LPARA;nCnt,X,Y,nWidt,
nHeigt:Int):BOOL;
12884: Function wInsertMenu( hMenu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar ) : BOOL
12885: //Function wInsertMenuItem( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo ) : BOOL
12886: Function wIsCharAlpha( ch : KOLChar ) : BOOL
12887: Function wIsCharAlphaNumeric( ch : KOLChar ) : BOOL
12888: Function wIsCharLower( ch : KOLChar ) : BOOL
12889: Function wIsCharUpper( ch : KOLChar ) : BOOL
12890: Function wIsDialogMessage( hDlg : HWND; var lpMsg : TMsg ) : BOOL
12891: Function wLoadAccelerators( hInstance : HINST; lpTableName : PKOLChar ) : HACCEL
12892: Function wLoadBitmap( hInstance : HINST; lpBitmapName : PKOLChar ) : HBITMAP
12893: Function wLoadCursor( hInstance : HINST; lpCursorName : PKOLChar ) : HCURSOR
12894: Function wLoadCursorFromFile( lpFileName : PKOLChar ) : HCURSOR
12895: Function wLoadIcon( hInstance : HINST; lpIconName : PKOLChar ) : HICON
12896: Function wLoadImage(hInst:HINST;ImageName:PKOLChar;ImageType:UINT;X,Y:Integer;Flags:UINT): THandle
12897: Function wLoadKeyboardLayout( pwszKLID : PKOLChar; Flags : UINT ) : HKL
12898: Function wLoadMenu( hInstance : HINST; lpMenuName : PKOLChar ) : HMENU
12899: //Function wLoadMenuIndirect( lpMenuTemplate : Pointer ) : HMENU
12900: Function wLoadString(hInstance:HINST;uID: UINT; lpBuffer : PKOLChar;nBufferMax:Integer):Integer
12901: Function wMapVirtualKey( uCode, uMapType : UINT ) : UINT
12902: Function wMapVirtualKeyEx( uCode, uMapType : UINT; dwHkl : HKL ) : UINT
12903: Function wMessageBox( hWnd : HWND; lpText, lpCaption : PKOLChar; uType : UINT ) : Integer
12904: Function wMessageBoxEx( hWnd:HWND; lpText,lpCaption:PKOLChar;uType:UINT;wLanguageId:Word): Integer
12905: //Function wMessageBoxIndirect( const MsgBoxParams : TMsgBoxParams ) : BOOL
12906: Function wModifyMenu( hMnu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar ) : BOOL
12907: //Function wOemToAnsi( const lpszSrc : LPCSTR; lpszDst : LPSTR ) : BOOL
12908: //7Function wOemToAnsiBuff( lpszSrc : LPCSTR; lpszDst : LPSTR; cchDstLength : DWORD ) : BOOL
12909: //Function wOemToChar( lpszSrc : PKOLChar; lpszDst : PKOLChar ) : BOOL
12910: Function wOemToCharBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD ) : BOOL
12911: Function wOpenDesktop(lpszDesktop:PKOLChar;dwFlags:DWORD;fInherit:BOOL;dwDesiredAccess:DWORD): HDESK
12912: Function wOpenWindowStation( lpszWinSta : PKOLChar; fInherit : BOOL; dwDesiredAccess : DWORD ) : HWINSTA
12913: Function wPeekMessage( var lpMsg : TMsg; hWnd : HWND; wParam:FilterMin,wParam:FilterMax, wRemoveMsg:UINT):BOOL
12914: Function wPostMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
12915: Function wPostThreadMessage(idThread:DWORD;Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
12916: Function wRealGetWindowClass( hWnd : HWND; pszType : PKOLChar; cchType : UINT ) : UINT
12917: // Function wRegisterClass( const lpWndClass : TWndClass ) : ATOM
12918: // Function wRegisterClassEx( const WndClass : TWndClassEx ) : ATOM
12919: Function wRegisterClipboardFormat( lpszFormat : PKOLChar ) : UINT
12920: // Function wRegisterDeviceNotification(hRecipient:THandle;NotificFilter:Pointer;Flags:DWORD):HDEVNOTIFY

```

```

12921: Function wRegisterWindowMessage( lpString : PKOLChar ) : UINT
12922: Function wRemoveProp( hWnd : HWND; lpString : PKOLChar ) : THandle
12923: Function wSendDlgItemMessage( hDlg:HWND;nIDDlgItem:Integer;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
12924: Function wSendMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT
12925: //Function wSendMessageCallback( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM;
lpResultCallBack : TFNSendAsyncProc; dwData : DWORD ) : BOOL
12926: Function wSendMessageTimeout( hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM; fuFlags,uTimeout:UINT;var
lpdwResult:DWORD): LRESULT
12927: Function wSendNotifyMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
12928: Function wSetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : DWORD
12929: Function wSetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PKOLChar ) : BOOL
12930: //Function wSetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo ) : BOOL
12931: Function wSetProp( hWnd : HWND; lpString : PKOLChar; hData : THandle ) : BOOL
12932: // Function wSetUserObjectInformation(hObj:THandle;nIndex:Integer;pvInfo:Pointer;nLength:DWORD):BOOL
12933: Function wSetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : Longint
12934: Function wSetWindowText( hWnd : HWND; lpString : PKOLChar ) : BOOL
12935: //Function wSetWindowsHook( nFilterType : Integer; pfnFilterProc : TFNHookProc ) : HHOOK
12936: //Function wSetWindowsHookEx(idHook:Integer;lpfn:TFNHookProc;hmod:HINST;dwThreadId:DWORD):HHOOK;
12937: // Function wSystemParametersInfo( uiAction, uiParam : UINT; pvParam : Pointer; fWinIni: UINT):BOOL
12938: Function wTabbedTextOut( hDC:HDC;X,Y:Int;lpString:PKOLChar;nCount,nTabPositions:Int;var
lpnTabStopPositions,nTabOrigin:Int):Longint;
12939: Function wTranslateAccelerator( hWnd : HWND; hAccTable : HACCEL; var lpMsg : TMsg ) : Integer
12940: Function wUnregisterClass( lpClassName : PKOLChar; hInstance : HINST ) : BOOL
12941: Function wVkKeyScan( ch : KOLChar ) : SHORT
12942: Function wVkKeyScanEx( ch : KOLChar; dwHkl : HKL ) : SHORT
12943: Function wWinHelp( hWndMain : HWND; lpszHelp : PKOLChar; uCommand : UINT; dwData : DWORD ) : BOOL
12944: Function wwprintf( Output : PKOLChar; Format : PKOLChar ) : Integer
12945: Function wvwsprintf( Output : PKOLChar; Format : PKOLChar; arglist : va_list ) : Integer
12946:
12947: //TestDrive!
12948: 'SID_REVISION','LongInt').SetInt(1;'FILENAME_ADVAPI32','String').SetString('ADVAPI32.DLL
12949: 'PROC_CONVERTSIDTOSTRINGSIDA','String').SetString('ConvertSidToStringSida
12950: Function GetDomainUserSid(const domainName:String;const userName:String; var foundDomain:String):String;
12951: Function GetLocalUserSidStr( const UserName : string ) : string
12952: Function GetPid4user( const domain : string; const user : string; var pid : dword ) : boolean
12953: Function Impersonate2User( const domain : string; const user : string ) : boolean
12954: Function GetProcessUserByPid( pid : DWORD; var UserName, Domain : AnsiString ) : Boolean
12955: Function KillProcessbyname( const exename : string; var found : integer ) : integer
12956: Function getWinProcessList : TStringList
12957: end;
12958:
12959: procedure SIRegister_AfSafeSync(CL: TPSPascalCompiler);
12960: begin
12961: 'AfMaxSyncSlots','LongInt').SetInt( 64);
12962: 'AfSynchronizeTimeout','LongInt').SetInt( 2000);
12963: 'TafSyncSlotID', 'DWORD
12964: 'TafSyncStatistics','record MessagesCount:Int;TimeoutMessages:Int;DisabledMessages:Int;end;
12965: 'TafSafeSyncEvent','Procedure ( ID : TafSyncSlotID)
12966: 'TafSafeDirectSyncEvent','Procedure
12967: Function AfNewSyncSlot( const AEvent : TafSafeSyncEvent ) : TafSyncSlotID
12968: Function AfReleaseSyncSlot( const ID : TafSyncSlotID ) : Boolean
12969: Function AfEnableSyncSlot( const ID : TafSyncSlotID; Enable : Boolean ) : Boolean
12970: Function AfValidateSyncSlot( const ID : TafSyncSlotID ) : Boolean
12971: Function AfSyncEvent( const ID : TafSyncSlotID; Timeout : DWORD ) : Boolean
12972: Function AfDirectSyncEvent( Event : TafSafeDirectSyncEvent; Timeout : DWORD ) : Boolean
12973: Function AfIsSyncMethod : Boolean
12974: Function AfSyncWnd : HWND
12975: Function AfSyncStatistics : TafSyncStatistics
12976: Procedure AfClearSyncStatistics
12977: end;
12978:
12979: procedure SIRegister_AfComPortCore(CL: TPSPascalCompiler);
12980: begin
12981: 'fBinary','LongWord').SetUInt( $00000001);
12982: 'fParity','LongWord').SetUInt( $00000002);
12983: 'fOutxCtsFlow','LongWord').SetUInt( $00000004);
12984: 'fOutxDsrFlow','LongWord').SetUInt( $00000008);
12985: 'fDtrControl','LongWord').SetUInt( $00000030);
12986: 'fDtrControlDisable','LongWord').SetUInt( $00000000);
12987: 'fDtrControlEnable','LongWord').SetUInt( $00000010);
12988: 'fDtrControlHandshake','LongWord').SetUInt( $00000020);
12989: 'fDsrSensitivity','LongWord').SetUInt( $00000040);
12990: 'fTXContinueOnXoff','LongWord').SetUInt( $00000080);
12991: 'fOutX','LongWord').SetUInt( $00000100);
12992: 'fInX','LongWord').SetUInt( $00000200);
12993: 'fErrorChar','LongWord').SetUInt( $00000400);
12994: 'fNull','LongWord').SetUInt( $00000800);
12995: 'fRtsControl','LongWord').SetUInt( $00003000);
12996: 'fRtsControlDisable','LongWord').SetUInt( $00000000);
12997: 'fRtsControlEnable','LongWord').SetUInt( $00001000);
12998: 'fRtsControlHandshake','LongWord').SetUInt( $00002000);
12999: 'fRtsControlToggle','LongWord').SetUInt( $00003000);
13000: 'fAbortOnError','LongWord').SetUInt( $00004000);
13001: 'fDummy2','LongWord').SetUInt( $FFFF8000);
13002: 'TafCoreEvent', '( ceOutFree, ceLineEvent, ceNeedReadData, ceException )
13003: FindClass('TOBJECT'),'EAfComPortCoreError
13004: FindClass('TOBJECT'),'TafComPortCore
13005: 'TafComPortCoreEvent', 'Procedure ( Sender : TafComPortCore; Even'
13006: +tKind : TafCoreEvent; Data : DWORD)

```

```

13007:   SIRegister_TafComPortCoreThread(CL);
13008:   SIRegister_TafComPortEventThread(CL);
13009:   SIRegister_TafComPortWriteThread(CL);
13010:   SIRegister_TafComPortCore(CL);
13011:   Function FormatDeviceName( PortNumber : Integer ) : string
13012: end;
13013:
13014: procedure SIRegister_ApplicationFileIO(CL: TPSPascalCompiler);
13015: begin
13016:   TAFIOFileStreamEvent', 'Function ( const fileName : String; mode: Word ) : TStream
13017:   TAFIOFileStreamExistsEvent', 'Function ( const fileName : String ) : Boolean
13018:   SIRegister_TApplicationFileIO(CL);
13019:   TDataFileCapability', '( dfcRead, dfcWrite )
13020:   TDataFileCapabilities', 'set of TDataFileCapability
13021:   SIRegister_TDataFile(CL);
13022:   //TDataFileClass', 'class of TDataFile
13023:   Function ApplicationFileIODefined : Boolean
13024:   Function CreateFileStream(const fileName: String; mode: WordfmShareDenyNone):TStream
13025:   Function FileStreamExists(const fileName: String) : Boolean
13026:   //Procedure Register
13027: end;
13028:
13029: procedure SIRegister_ALFBXLib(CL: TPSPascalCompiler);
13030: begin
13031:   TALFBXFieldType', '( uftUnknown, uftNumeric, uftChar, uftVarchar'
13032:   +', uftCString, uftSmallint, uftInteger, uftQuad, uftFloat, uftDoublePrecisi'
13033:   +', on, uftTimestamp, uftBlob, uftBlobId, uftDate, uftTime, uftInt64, uftArray, uftNull )
13034:   TALFBXScale', 'Integer
13035:   FindClass('TOBJECT'),'EALFBXConvertError
13036:   SIRegister_EALFBXError(CL);
13037:   SIRegister_EALFBXException(CL);
13038:   FindClass('TOBJECT'),'EALFBXGFixError
13039:   FindClass('TOBJECT'),'EALFBXDSQLError
13040:   FindClass('TOBJECT'),'EALFBXDynError
13041:   FindClass('TOBJECT'),'EALFBXGBakError
13042:   FindClass('TOBJECT'),'EALFBXGSecError
13043:   FindClass('TOBJECT'),'EALFBXLicenseError
13044:   FindClass('TOBJECT'),'EALFBXGStatError
13045:   //EALFBXExceptionClass', 'class of EALFBXError
13046:   TALFBXCharacterSet', '( csNONE, csASCII, csBIG_5, csCYRL, csDOS4'
13047:   +', csDOS850, csDOS852, csDOS857, csDOS860, csDOS861, csDOS863, csDOS865, '
13048:   +', csEUCJ_0208, csGB_2312, csISO8859_1, csISO8859_2, csKSC_5601, csNEXT, csOC'
13049:   +', TETS, csSJIS_0208, csUNICODE_FSS, csUTF8, csWIN1250, csWIN1251, csWIN1252,'
13050:   +', csWIN1253, csWIN1254, csDOS737, csDOS775, csDOS858, csDOS862, csDOS864, c'
13051:   +', sDOS866, csDOS869, csWIN1255, csWIN1256, csWIN1257, csISO8859_3, csISO8859'
13052:   +', _4, csISO8859_5, csISO8859_6, csISO8859_7, csISO8859_8, csISO8859_9, csISO'
13053:   +', 8859_13, csKOI8R, csKOI8U, csWIN1258, csTIS620, csGBK, csCP943C )
13054:   TALFBXTransParam', '( tpConsistency, tpConcurrency, tpShared, tp'
13055:   +', Protected, tpExclusive, tpWait, tpNowait, tpRead, tpWrite, tpLockRead, tpL'
13056:   +', LockWrite, tpVerbTime, tpCommitTime, tpIgnoreLimbo, tpReadCommitted, tpAuto'
13057:   +', Commit, tpRecVersion, tpNoRecVersion, tpRestartRequests, tpNoAutoUndo, tpLockTimeout )
13058:   TALFBXTransParams', 'set of TALFBXTransParam
13059:   Function ALFBXStrToCharacterSet( const CharacterSet : AnsiString ) : TALFBXCharacterSet
13060:   Function ALFBXCreateDBParams( Params : AnsiString; Delimiter : Char ) : AnsiString
13061:   Function ALFBXCreateBlobParams( Params : AnsiString; Delimiter : Char ) : AnsiString
13062:   'cALFBXMaxParamLength', 'LongInt'.SetInt( 125);
13063:   TALFBXParamsFlag', '( pfNotInitialized, pfNotNullable )
13064:   TALFBXParamsFlags', 'set of TALFBXParamsFlag
13065:   //PALFBXSQLVar', '^TALFBXSQLVar // will not work
13066:   //PALFBXSQLDaData', '^TALFBXSQLDaData // will not work
13067:   TALFBXStatementType', '( stSelect, stInsert, stUpdate, stDelete,'
13068:   +', stDDL, stGetSegment, stPutSegment, stExecProcedure, stStartTrans, stComm'
13069:   +', t, stRollback, stSelectForUpdate, stSetGenerator, stSavePoint )
13070:   SIRegister_TALFBXSQLDA(CL);
13071:   //PALFBXPtrArray', '^TALFBXPtrArray // will not work
13072:   SIRegister_TALFBXPoolStream(CL);
13073:   //PALFBXBlobData', '^TALFBXBlobData // will not work
13074:   TALFBXBlobData', 'record Size : Integer; Buffer : string; end
13075:   //PALFBXArrayDesc', '^TALFBXArrayDesc // will not work
13076:   //TALFBXArrayDesc', 'TISCArrDesc
13077:   //TALFBXBlobDesc', 'TISCBlobDesc
13078:   //PALFBXArrayInfo', '^TALFBXArrayInfo // will not work
13079:   //TALFBXArrayInfo', 'record index : Integer; size : integer; info: TALFBXArrayDesc; end
13080:   SIRegister_TALFBXSQLResult(CL);
13081:   //TALFBXSQLResultClass', 'class of TALFBXSQLResult
13082:   SIRegister_TALFBXSQLParams(CL);
13083:   //TALFBXSQLParamsClass', 'class of TALFBXSQLParams
13084:   TALFBXDSQInfoData', 'record InfoCode : byte; InfoLen : Word; St'
13085:   +', atementType : TALFBXStatementType; end
13086:   FindClass('TOBJECT'),'TALFBXLibrary
13087:   //PALFBXStatusVector', '^TALFBXStatusVector // will not work
13088:   TALFBXOnConnectionLost', 'Procedure ( Lib : TALFBXLibrary)
13089:   //TALFBXOnGetDBExceptionClass', 'Procedure ( Number : Integer; out'
13090:   //'+ Excep : EALFBXExceptionClass)
13091:   SIRegister_TALFBXLibrary(CL);
13092:   'cALFBXDateOffset', 'LongInt'.SetInt( 15018);
13093:   'cALFBXTimeCoeff', 'LongInt'.SetInt( 864000000);
13094:   //Procedure ALFBXDecodeTimeStamp( v : PISCTimeStamp; out DateTime : Double);
13095:   //Procedure ALFBXDecodeTimeStamp1( v : PISCTimeStamp; out TimeStamp : TTimeStamp);

```



```

13096: //Function ALFBXDecodeTimeStamp( v : PISCTimeStamp ) : Double;
13097: Procedure ALFBXDecodeSQLDate( v : Integer; out Year : SmallInt; out Month, Day : Word)
13098: Procedure ALFBXDecodeSQLTime(v:Cardinal;out Hour,Minute,Second:Word; out Fractions: LongWord)
13099: //Procedure ALFBXEncodeTimeStamp( const DateTime : TDateTime; v : PISCTimeStamp);
13100: //Procedure ALFBXEncodeTimeStamp1( const Date : Integer; v : PISCTimeStamp);
13101: //Procedure ALFBXEncodeTimeStamp2( const Time : Cardinal; v : PISCTimeStamp);
13102: Function ALFBXEncodeSQLDate( Year : Integer; Month, Day : Integer ) : Integer
13103: Function ALFBXEncodeSQLTime( Hour, Minute, Second : Word; var Fractions : LongWord): Cardinal
13104: TALFBXParamType', '( prNone, prByte, prShrt, prCard, prStrg, prIgno )
13105: TALFBXDPPInfo', 'record Name : AnsiString; ParamType : TALFBXParamType; end
13106: Function ALFBXSQLQuote( const name : AnsiString ) : AnsiString
13107: Function ALFBXSQLUnQuote( const name : AnsiString ) : AnsiString
13108: end;
13109:
13110: procedure SIRegister_ALFBXClient(CL: TPSPascalCompiler);
13111: begin
13112: TALFBXClientSQLParam', 'record Value : AnsiString; IsNull : Boolean; end
13113: TALFBXClientSQLParams', 'array of TALFBXClientSQLParam
13114: TALFBXClientSelectDataSQL', 'record SQL : AnsiString; Params : T'
13115: + 'ALFBXClientSQLParams; RowTag : AnsiString; ViewTag : AnsiString; Skip : in'
13116: + 'eger; First : Integer; CacheThreshold : Integer; end
13117: TALFBXClientSelectDataSQLs', 'array of TALFBXClientSelectDataSQL
13118: TALFBXClientUpdateDataSQL', 'record SQL : AnsiString; Params: TALFBXClientSQLParams; end
13119: TALFBXClientUpdateDataSQLs', 'array of TALFBXClientUpdateDataSQL
13120: TALFBXClientMonitoringIOStats', 'record page_reads : int64; page'
13121: + '_writes : int64; page_fetches : int64; page_marks : int64; end
13122: SIRegister_TALFBXClient(CL);
13123: SIRegister_TALFBXConnectionStatementPoolBinTreeNode(CL);
13124: SIRegister_TALFBXConnectionStatementPoolBinTree(CL);
13125: SIRegister_TALFBXConnectionWithStmntPoolContainer(CL);
13126: SIRegister_TALFBXConnectionWithoutStmntPoolContainer(CL);
13127: SIRegister_TALFBXReadTransactionPoolContainer(CL);
13128: SIRegister_TALFBXReadStatementPoolContainer(CL);
13129: SIRegister_TALFBXStringKeyPoolBinTreeNode(CL);
13130: SIRegister_TALFBXConnectionPoolClient(CL);
13131: SIRegister_TALFBXEventThread(CL);
13132: Function AlMySQLClientSlashedStr( const Str : AnsiString ) : AnsiString
13133: end;
13134:
13135: procedure SIRegister_ovcBidi(CL: TPSPascalCompiler);
13136: begin
13137: _OSVERSIONINFOA = record
13138: dwOSVersionInfoSize: DWORD;
13139: dwMajorVersion: DWORD;
13140: dwMinorVersion: DWORD;
13141: dwBuildNumber: DWORD;
13142: dwPlatformId: DWORD;
13143: szCSDVersion: array[0..127] of AnsiChar; { Maintenance AnsiString for PSS usage }
13144: end;
13145: TOSVersionInfoA', '_OSVERSIONINFOA
13146: TOSVersionInfo', 'TOSVersionInfoA
13147: 'WS_EX_RIGHT','LongWord').SetUInt( $00001000);
13148: 'WS_EX_LEFT','LongWord').SetUInt( $00000000);
13149: 'WS_EX_RTLREADING','LongWord').SetUInt( $00002000);
13150: 'WS_EX_LTRREADING','LongWord').SetUInt( $00000000);
13151: 'WS_EX_LEFTSCROLLBAR','LongWord').SetUInt( $00004000);
13152: 'WS_EX_RIGHTSCROLLBAR','LongWord').SetUInt( $00000000);
13153: Function SetProcessDefaultLayout( dwDefaultLayout : DWORD ) : BOOL
13154: 'LAYOUT_RTL','LongWord').SetUInt( $00000001);
13155: 'LAYOUT_BTT','LongWord').SetUInt( $00000002);
13156: 'LAYOUT_VBTH','LongWord').SetUInt( $00000004);
13157: 'LAYOUT_BITMAPORIENTATIONPRESERVED','LongWord').SetUInt( $00000008);
13158: 'NOMIRRORBITMAP','LongWord').SetUInt( DWORD ( $80000000 ));
13159: Function SetLayout( dc : HDC; dwLayout : DWORD ) : DWORD
13160: Function GetLayout( dc : hdc ) : DWORD
13161: Function IsBidi : Boolean
13162: Function GetCurrentHwProfile( var lpHwProfileInfo : THWProfileInfo ) : BOOL
13163: Function GetVersionEx( var lpVersionInformation : TOSVersionInfo ) : BOOL
13164: Function SetPriorityClass( hProcess : THandle; dwPriorityClass: DWORD ) : BOOL
13165: Function GetPriorityClass( hProcess : THandle ) : DWORD
13166: Function OpenClipboard( hWndNewOwner : HWND ) : BOOL
13167: Function CloseClipboard : BOOL
13168: Function GetClipboardSequenceNumber : DWORD
13169: Function GetClipboardOwner : HWND
13170: Function SetClipboardViewer( hWndNewViewer : HWND ) : HWND
13171: Function GetClipboardViewer : HWND
13172: Function ChangeClipboardChain( hWndRemove, hWndNewNext : HWND ) : BOOL
13173: Function SetClipboardData( uFormat : UINT; hMem : THandle ) : THandle
13174: Function GetClipboardData( uFormat : UINT ) : THandle
13175: Function RegisterClipboardFormat( lpszFormat : PChar ) : UINT
13176: Function CountClipboardFormats : Integer
13177: Function EnumClipboardFormats( format : UINT ) : UINT
13178: Function GetClipboardFormatName(format:UINT;lpszFormatName:PChar;cchMaxCount:Integer):Integer
13179: Function EmptyClipboard : BOOL
13180: Function IsClipboardFormatAvailable( format : UINT ) : BOOL
13181: Function GetPriorityClipboardFormat( var paFormatPriorityList, cFormats : Integer ) : Integer
13182: Function GetOpenClipboardWindow : HWND
13183: Function EndDialog( hDlg : HWND; nResult : Integer ) : BOOL
13184: Function GetDlgItem( hDlg : HWND; nIDlgItem : Integer ) : HWND

```

```

13185: Function SetDlgItemInt( hDlg : HWND; nIDDlgItem : Integer; uValue : UINT; bSigned: BOOL): BOOL
13186: Function GetDlgItemInt(hDlg:HWND;nIDDlgItem:Integer;var lpTranslated:BOOL;bSigned: BOOL): UINT
13187: Function SetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PChar) : BOOL
13188: Function CheckDlgButton( hDlg : HWND; nIDButton : Integer; uCheck : UINT) : BOOL
13189: Function CheckRadioButton( hDlg : HWND; nIDFirstButton, nIDLastButton,nIDCheckButton:Integer) : BOOL
13190: Function IsDlgButtonChecked( hDlg : HWND; nIDButton : Integer) : UINT
13191: Function SendDlgItemMessage(hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13192: end;
13193:
13194: procedure SIRegister_DXPUtils(CL: TPSPascalCompiler);
13195: begin
13196:   Function glExecuteAndWait(cmdLine:String;visibility:Word;timeout:Cardinal;killAppOnTimeOut:Bool):Int;
13197:   Function GetTemporaryFilePath : String
13198:   Function GetTemporaryFileName : String
13199:   Function FindFileInPaths( const fileName, paths : String) : String
13200:   Function PathsToString( const paths : TStringList) : String
13201:   Procedure StringToPaths( const pathsString : String; paths : TStringList)
13202:   //Function MacroExpandPath( const aPath : String) : String
13203: end;
13204:
13205: procedure SIRegister_ALMultiPartBaseParser(CL: TPSPascalCompiler);
13206: begin
13207:   SIRegister_TALMultiPartBaseContent(CL);
13208:   SIRegister_TALMultiPartBaseContents(CL);
13209:   SIRegister_TALMultiPartBaseStream(CL);
13210:   SIRegister_TALMultiPartBaseEncoder(CL);
13211:   SIRegister_TALMultiPartBaseDecoder(CL);
13212:   Function ALMultiPartExtractBoundaryFromContentType( aContentType : AnsiString) : AnsiString
13213:   Function ALMultiPartExtractSubValueFromHeaderLine(aHeaderLine:AnsiString;aName:AnsiString):AnsiString;
13214:   Function ALMultiPartSetSubValueInHeaderLine(aHeaderLine:AnsiString;aName,AValue:AnsiString):AnsiString;
13215: end;
13216:
13217: procedure SIRegister_SmallUtils(CL: TPSPascalCompiler);
13218: begin
13219:   TdriveSize', 'record FreeS : Int64; Totals : Int64; end
13220:   TWinVerRec', 'record WinPlatform : Integer; WinMajorVersion : In
13221:   +teger; WinMinorVersion :Integer; WinBuildNumber : Integer; WinCSDVersion: String; end
13222:   Function aAllocPadedMem( Size : Cardinal) : TObjct
13223:   Procedure aFreePadedMem( var P : TObjct);
13224:   Procedure aFreePadedMeml( var P : PChar);
13225:   Function aCheckPadedMem( P : Pointer) : Byte
13226:   Function aGetPadMemSize( P : Pointer) : Cardinal
13227:   Function aAllocMem( Size : Cardinal) : Pointer
13228:   Function aStrLen( const Str : PChar) : Cardinal
13229:   Function aStrCopy( Dest : PChar; const Source : PChar; MaxLen : Cardinal) : PChar
13230:   Function aStrECopy( Dest : PChar; const Source : PChar) : PChar
13231:   Function aStrCopy( Dest : PChar; const Source : PChar) : PChar
13232:   Function aStrEnd( const Str : PChar) : PChar
13233:   Function aStrScan( const Str : PChar; aChr : Char) : PChar
13234:   Function aStrMove( Dest : PChar; const Source : PChar; Count : Cardinal) : PChar
13235:   Function aPCharLength( const Str : PChar) : Cardinal
13236:   Function aPCharUpper( Str : PChar) : PChar
13237:   Function aPCharLower( Str : PChar) : PChar
13238:   Function aStrCat( Dest : PChar; const Source : PChar) : PChar
13239:   Function aLastDelimiter( const Delimiters, S : String) : Integer
13240:   Function aCopyTail( const S : String; Len : Integer) : String
13241:   Function aInt2Thos( I : Int64) : String
13242:   Function aUpperCase( const S : String) : String
13243:   Function aLowerCase( const S : string) : String
13244:   Function aCompareText( const S1, S2 : string) : Integer
13245:   Function aSameText( const S1, S2 : string) : Boolean
13246:   Function aInt2Str( Value : Int64) : String
13247:   Function aStr2Int( const Value : String) : Int64
13248:   Function aStr2IntDef( const S : string; Default : Int64) : Int64
13249:   Function aGetFileExt( const FileName : String) : String
13250:   Function aGetFilePath( const FileName : String) : String
13251:   Function aGetFileName( const FileName : String) : String
13252:   Function aChangeExt( const FileName, Extension : String) : String
13253:   Function aAdjustLineBreaks( const S : string) : string
13254:   Function aGetWindowStr( WinHandle : HWND) : String
13255:   Function aDiskSpace( Drive : String) : TdriveSize
13256:   Function aFileExists( FileName : String) : Boolean
13257:   Function aFileSize( FileName : String) : Int64
13258:   Function aDirectoryExists( const Name : string) : Boolean
13259:   Function aSysErrorMessage( ErrorCode : Integer) : string
13260:   Function aShortPathName( const LongName : string) : string
13261:   Function aGetWindowVer : TWinVerRec
13262:   procedure InitDriveSpacePtr;
13263: end;
13264:
13265: procedure SIRegister_MakeApp(CL: TPSPascalCompiler);
13266: begin
13267:   aZero', 'LongInt').SetInt( 0);
13268:   'makeappDEF', 'LongInt').SetInt( - 1);
13269:   'CS_VREDRAW', 'LongInt').SetInt( DWORD ( 1 ));
13270:   'CS_HREDRAW', 'LongInt').SetInt( DWORD ( 2 ));
13271:   'CS_KEYCVTWINDOW', 'LongInt').SetInt( 4);
13272:   'CS_DBLCLKS', 'LongInt').SetInt( 8);
13273:   'CS_OWNDC', 'LongWord').SetUInt( $20);

```

```

13274: 'CS_CLASSDC','LongWord').SetUInt( $40);
13275: 'CS_PARENTDC','LongWord').SetUInt( $80);
13276: 'CS_NOKEYCVT','LongWord').SetUInt( $100);
13277: 'CS_NOCLOSE','LongWord').SetUInt( $200);
13278: 'CS_SAVEBITS','LongWord').SetUInt( $800);
13279: 'CS_BYTEALIGNCLIENT','LongWord').SetUInt( $1000);
13280: 'CS_BYTEALIGNWINDOW','LongWord').SetUInt( $2000);
13281: 'CS_GLOBALCLASS','LongWord').SetUInt( $4000);
13282: 'CS_IME','LongWord').SetUInt( $10000);
13283: 'CS_DROPSHADOW','LongWord').SetUInt( $20000);
13284: //PPanelFunc, '^TPanelFunc // will not work
13285: TPanelStyle', '(psEdge, psTabEdge, psBorder, psTabBorder, psTab, psNone )
13286: TFontLook', '( flBold, flItalic, flUnderLine, flStrikeOut )
13287: TFontLooks', 'set of TFontLook
13288: TMessagefunc', 'function(hWnd,iMsg,wParam,lParam:Integer):Integer)
13289: Function SetWinClass(const ClassName:String; pMessFunc: TMessagefunc; wcStyle : Integer): Word
13290: Function SetWinClassO( const ClassName : String; pMessFunc : TObject; wcStyle : Integer): Word
13291: Function SetWinClass( const ClassName : String; pMessFunc : TObject; wcStyle : Integer) : Word
13292: Function MakeForm(Left,Top,Width,Height:Integer;const Caption:String;WinStyle:Integer):Integer
13293: Procedure RunMsgLoop( Show : Boolean)
13294: Function MakeFont(Height,Width:Integer; const FontName:String; Look:TFontLooks; Roman:Boolean): Integer
13295: Function MakeButton(Left,Top,Width,Height:Integer;pCaption:PChar;hParent,
ID_Number:Cardinal/hFont:Int):Int;
13296: Function MakeListBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Integer
13297: Function MakeComboBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Int
13298: Function MakePanel(Left,Top,Width,Height,
hParent:Int;WndFunc:TPanelFunc;ID_Number:Card;Style:TPanelStyle):Int;
13299: Function MakeSubMenu(const ItemList : String; ID1, ID2 : Cardinal; hMenu : Integer) : Integer
13300: Function id4menu( a, b : Byte; c : Byte; d : Byte) : Cardinal
13301: Procedure DoInitMakeApp //set first to init formclasscontrol!
13302: end;
13303:
13304: procedure SIRegister_ScreenSaver(CL: TPSPascalCompiler);
13305: begin
13306:   TScreenSaverOption', '( ssoAutoAdjustFormProperties, ssoAutoHook'
13307:   + 'KeyboardEvents, ssoAutoHookMouseEvents, ssoEnhancedMouseMoveDetection )
13308:   TScreenSaverOptions', 'set of TScreenSaverOption
13309:   'cDefaultScreenSaverOptions','LongInt').Value.ts32:=ord(ssoAutoAdjustFormProperties) or
ord(ssoAutoHookKeyboardEvents) or ord(ssoEnhancedMouseMoveDetection);
13310:   TScreenSaverPreviewEvent', 'Procedure ( Sender : TObject; previewHwnd: HWND)
13311:   SIRegister_TScreenSaver(CL);
13312:   //Procedure Register
13313:   Procedure SetScreenSaverPassword
13314: end;
13315:
13316: procedure SIRegister_XCollection(CL: TPSPascalCompiler);
13317: begin
13318:   FindClass('TObject'),'TXCollection
13319:   SIRegister_EFilerException(CL);
13320:   SIRegister_TXCollectionItem(CL);
13321:   //TXCollectionItemClass', 'class of TXCollectionItem
13322:   SIRegister_TXCollection(CL);
13323:   Procedure RegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13324:   Procedure DeRegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13325:   Procedure RegisterXCollectionItemClass( aclass : TXCollectionItemClass)
13326:   Procedure UnregisterXCollectionItemClass( aClass : TXCollectionItemClass)
13327:   Function FindXCollectionItemClass( const className : String) : TXCollectionItemClass
13328:   Function GetXCollectionItemClassesList( baseClass : TXCollectionItemClass) : TList
13329: end;
13330:
13331: procedure SIRegister_XOpenGL(CL: TPSPascalCompiler);
13332: begin
13333:   TMapTexCoordMode', '( mtcUndefined, mtcNull, mtcMain, mtcDual, mtcSecond,mtcmArbitrary);
13334:   Procedure xglMapTexCoordToNull
13335:   Procedure xglMapTexCoordToMain
13336:   Procedure xglMapTexCoordToSecond
13337:   Procedure xglMapTexCoordToDual
13338:   Procedure xglMapTexCoordToArbitrary( const units : array of Cardinal);
13339:   Procedure xglMapTexCoordToArbitrary1( const bitWiseUnits : Cardinal);
13340:   Procedure xglMapTexCoordToArbitraryAdd( const bitWiseUnits : Cardinal)
13341:   Procedure xglBeginUpdate
13342:   Procedure xglEndUpdate
13343:   Procedure xglPushState
13344:   Procedure xglPopState
13345:   Procedure xglForbidSecondTextureUnit
13346:   Procedure xglAllowSecondTextureUnit
13347:   Function xglGetBitWiseMapping : Cardinal
13348: end;
13349:
13350: procedure SIRegister_VectorLists(CL: TPSPascalCompiler);
13351: begin
13352:   TBaseListOption', '( bloExternalMemory, bloSetCountResetsMemory)
13353:   TBaseListOptions', 'set of TBaseListOption
13354:   SIRegister_TBaseList(CL);
13355:   SIRegister_TBaseVectorList(CL);
13356:   SIRegister_TAffineVectorList(CL);
13357:   SIRegister_TVVectorList(CL);
13358:   SIRegister_TTexPointList(CL);
13359:   SIRegister_TXIntegerList(CL);

```

```

13360: //PSingleArrayList', '^TSingleArrayList // will not work
13361: SIRegister_TSingleList(CL);
13362: SIRegister_TByteList(CL);
13363: SIRegister_TQuaternionList(CL);
13364: Procedure QuickSortLists( startIndex, endIndex : Integer; refList : TSingleList; objList : TList);
13365: Procedure QuickSortLists1( startIndex, endIndex : Integer; refList : TSingleList; objList : TBaseList);
13366: Procedure FastQuickSortLists(startIndex,
endIndex:Integer;refList:TSingleList;objList:TPersistentObjectList);
13367: end;
13368:
13369: procedure SIRegister_MeshUtils(CL: TPSPascalCompiler);
13370: begin
13371: Procedure ConvertStripToTList( const strip : TAffineVectorList; list : TAffineVectorList);
13372: Procedure ConvertStripToTList1( const strip : TIntegerList; list : TIntegerList);
13373: Procedure ConvertStripToTList2(const strip:TAffineVectorList;const
indices:TIntegerList;list:TAffineVectorList);
13374: Procedure ConvertIndexedTListToTList(const data:TAffineVectlist;const
indices:TIntegerList;list:TAffineVectorList);
13375: Function BuildVectorCountOptimizedIndices(const vertices:TAffineVectorList; const
normals:TAffineVectorList; const texCoords : TAffineVectorList) : TIntegerList
13376: Procedure RemapReferences( reference : TAffineVectorList; const indices : TIntegerList);
13377: Procedure RemapReferences1( reference : TIntegerList; const indices : TIntegerList);
13378: Procedure RemapAndCleanupReferences( reference : TAffineVectorList; indices : TIntegerList)
13379: Function RemapIndicesToIndicesMap( remapIndices : TIntegerList) : TIntegerList
13380: Procedure RemapTrianglesIndices( indices, indicesMap : TIntegerList)
13381: Procedure RemapIndices( indices, indicesMap : TIntegerList)
13382: Procedure UnifyTrianglesWinding( indices : TIntegerList)
13383: Procedure InvertTrianglesWinding( indices : TIntegerList)
13384: Function BuildNormals( reference : TAffineVectorList; indices : TIntegerList) : TAffineVectorList
13385: Function BuildNonOrientedEdgesList(triangleIndices:TIntegerList; triangleEdges : TIntegerList;
edgesTriangles : TIntegerList) : TIntegerList
13386: Procedure WeldVertices( vertices : TAffineVectorList; indicesMap : TIntegerList; weldRadius : Single)
13387: Function StripifyMesh(indices:TIntegerList;maxVertexIndex:Integer;agglomerateLoneTriangles:Boolean):
TPersistentObjectList;
13388: Procedure IncreaseCoherency( indices : TIntegerList; cacheSize : Integer)
13389: Procedure SubdivideTriangles( smoothFactor : Single; vertices : TAffineVectorList; triangleIndices :
TIntegerList; normals : TAffineVectorList; onSubdivideEdge : TSubdivideEdgeEvent)
13390: end;
13391:
13392: procedure SIRegister_JclSysUtils(CL: TPSPascalCompiler);
13393: begin
13394: Procedure GetAndFillMem( var P : TObject; const Size : Integer; const Value : Byte)
13395: Procedure FreeMemAndNil( var P : TObject)
13396: Function PCharOrNil( const S : string) : PChar
13397: SIRegister_TJclReferenceMemoryStream(CL);
13398: FindClass('TObject','EJclVMError
13399: {Function GetVirtualMethodCount( AClass : TClass) : Integer
13400: Function GetVirtualMethod( AClass : TClass; const Index : Integer) : Pointer
13401: Procedure SetVirtualMethod( AClass : TClass; const Index : Integer; const Method:Pointer)
13402: PDynamicIndexList', '^TDynamicIndexList // will not work
13403: PDynamicAddressList', '^TDynamicAddressList // will not work
13404: Function GetDynamicMethodCount( AClass : TClass) : Integer
13405: Function GetDynamicIndexList( AClass : TClass) : PDynamicIndexList
13406: Function GetDynamicAddressList( AClass : TClass) : PDynamicAddressList
13407: Function HasDynamicMethod( AClass : TClass; Index : Integer) : Boolean
13408: Function GetDynamicMethod( AClass : TClass; Index : Integer) : Pointer
13409: Function GetInitTable( AClass : TClass) : PTypeInfo
13410: PFieldEntry', '^TFieldEntry // will not work}
13411: TFieldEntry', 'record OffSet : Integer; IDX : Word; Name : Short'
13412: + 'String; end
13413: Function JIsClass( Address : Pointer) : Boolean
13414: Function JIsObject( Address : Pointer) : Boolean
13415: Function GetImplementorOfInterface( const I : IInterface) : TObject
13416: TDigitCount', 'Integer
13417: SIRegister_TJclNumericFormat(CL);
13418: Function JIntToStrZeroPad( Value, Count : Integer) : AnsiString
13419: TTextHandler', 'Procedure ( const Text : string)
13420: // 'ABORT_EXIT_CODE','LongInt').SetInt( ERROR_CANCELLED 1223);
13421: Function JExecute(const
CommandLine:string;OutputLineCallback:TTextHandler;RawOutput:Bool;AbortPtr:PBool):Card;
13422: Function JExecute1(const CommandLine:string;var Output:string; RawOutput:Bool; AbortPtr:PBool):Cardinal;
13423: Function ReadKey : Char //to and from the DOS console !
13424: TModuleHandle', 'HINST
13425: //TModuleHandle', 'Pointer
13426: 'INVALID_MODULEHANDLE_VALUE','LongInt').SetInt( TModuleHandle ( 0 ));
13427: Function LoadModule( var Module : TModuleHandle; FileName : string) : Boolean
13428: Function LoadModuleEx( var Module : TModuleHandle; FileName : string; Flags : Cardinal) : Boolean
13429: Procedure UnloadModule( var Module : TModuleHandle)
13430: Function GetModuleSymbol( Module : TModuleHandle; SymbolName : string) : Pointer
13431: Function GetModuleSymbolEx( Module : TModuleHandle; SymbolName : string; var Accu : Boolean) : Pointer
13432: Function ReadModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size: Cardinal):Boolean;
13433: Function WriteModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size:Cardinal):Boolean;
13434: FindClass('TObject','EJclConversionError
13435: Function JStrToBoolean( const S : string) : Boolean
13436: Function JBooleanToStr( B : Boolean) : string
13437: Function JIntToBool( I : Integer) : Boolean
13438: Function JBoolToInt( B : Boolean) : Integer
13439: 'ListSeparator','String').SetString( '
13440: 'ListSeparator1','String').SetString( '

```



```
13441: Procedure ListAddItems( var List : string; const Separator, Items : string)
13442: Procedure ListIncludeItems( var List : string; const Separator, Items : string)
13443: Procedure ListRemoveItems( var List : string; const Separator, Items : string)
13444: Procedure ListDelItem( var List : string; const Separator : string; const Index : Integer)
13445: Function ListItemCount( const List, Separator : string) : Integer
13446: Function ListGetItem( const List, Separator : string; const Index : Integer) : string
13447: Procedure ListSetItem(var List:string;const Separator:string;const Index:Integer;const Value:string)
13448: Function ListItemIndex( const List, Separator, Item : string) : Integer
13449: Function SystemToObjectInstance : LongWord
13450: Function IsCompiledWithPackages : Boolean
13451: Function JjclGUIDToString( const GUID : TGUID) : string
13452: Function JjclStringToGUID( const S : string) : TGUID
13453: SIRegister_TJclIntfCriticalSection(CL);
13454: SIRegister_TJclSimpleLog(CL);
13455: Procedure InitSimpleLog( const ALogFileName : string)
13456: end;
13457:
13458: procedure SIRegister_JclBorlandTools(CL: TPSPascalCompiler);
13459: begin
13460:   FindClass('TOBJECT'),'EJclBorRADException
13461:   TJclBorRADToolKind', '( brDelphi, brCppBuilder, brBorlandDevStudio )
13462:   TJclBorRADToolEdition', '( deOPEN, dePRO, deSVR )
13463:   TJclBorRADToolEdition', '( deSTD, dePRO, deCSS, deARC )
13464:   TJclBorRADToolPath', 'string
13465:   'SupportedDelphiVersions','LongInt').SetInt( 5 or 6 or 7 or 8 or 9 or 10 or 11);
13466:   'SupportedBCBVersions','LongInt').SetInt( 5 or 6 or 10 or 11);
13467:   'SupportedBDSVersions','LongInt').SetInt( 1 or 2 or 3 or 4 or 5);
13468:   BorRADToolRepositoryPagesSection','String').SetString( 'Repository Pages
13469:   BorRADToolRepositoryDialogsPage','String').SetString( 'Dialogs
13470:   BorRADToolRepositoryFormsPage','String').SetString( 'Forms
13471:   BorRADToolRepositoryProjectsPage','String').SetString( 'Projects
13472:   BorRADToolRepositoryDataModulesPage','String').SetString( 'Data Modules
13473:   BorRADToolRepositoryObjectType','String').SetString( 'Type
13474:   BorRADToolRepositoryFormTemplate','String').SetString( 'FormTemplate
13475:   BorRADToolRepositoryProjectTemplate','String').SetString( 'ProjectTemplate
13476:   BorRADToolRepositoryObjectName','String').SetString( 'Name
13477:   BorRADToolRepositoryObjectPage','String').SetString( 'Page
13478:   BorRADToolRepositoryObjectIcon','String').SetString( 'Icon
13479:   BorRADToolRepositoryObjectDescr','String').SetString( 'Description
13480:   BorRADToolRepositoryObjectAuthor','String').SetString( 'Author
13481:   BorRADToolRepositoryObjectAncestor','String').SetString( 'Ancestor
13482:   BorRADToolRepositoryObjectDesigner','String').SetString( 'Designer
13483:   BorRADToolRepositoryDesignerDfm','String').SetString( 'dfm
13484:   BorRADToolRepositoryDesignerXfm','String').SetString( 'xfm
13485:   BorRADToolRepositoryObjectNewForm','String').SetString( 'DefaultNewForm
13486:   BorRADToolRepositoryObjectMainForm','String').SetString( 'DefaultMainForm
13487:   SourceExtensionDelphiPackage','String').SetString( '.dpk
13488:   SourceExtensionBCBPackage','String').SetString( '.bpk
13489:   SourceExtensionDelphiProject','String').SetString( '.dpr
13490:   SourceExtensionBCBProject','String').SetString( '.bpr
13491:   SourceExtensionBDSProject','String').SetString( '.bdsproj
13492:   SourceExtensionDProject','String').SetString( '.dproj
13493:   BinaryExtensionPackage','String').SetString( '.bpl
13494:   BinaryExtensionLibrary','String').SetString( '.dll
13495:   BinaryExtensionExecutable','String').SetString( '.exe
13496:   CompilerExtensionDCP','String').SetString( '.dcp
13497:   CompilerExtensionBPI','String').SetString( '.bpi
13498:   CompilerExtensionLIB','String').SetString( '.lib
13499:   CompilerExtensionTDS','String').SetString( '.tds
13500:   CompilerExtensionMAP','String').SetString( '.map
13501:   CompilerExtensionDRC','String').SetString( '.drc
13502:   CompilerExtensionDEF','String').SetString( '.def
13503:   SourceExtensionCPP','String').SetString( '.cpp
13504:   SourceExtensionH','String').SetString( '.h
13505:   SourceExtensionPAS','String').SetString( '.pas
13506:   SourceExtensionDFM','String').SetString( '.dfm
13507:   SourceExtensionXFM','String').SetString( '.xfm
13508:   SourceDescriptionPAS','String').SetString( 'Pascal source file
13509:   SourceDescriptionCPP','String').SetString( 'C++ source file
13510:   DesignerVCL','String').SetString( 'VCL
13511:   DesignerCLX','String').SetString( 'CLX
13512:   ProjectTypePackage','String').SetString( 'package
13513:   ProjectTypeLibrary','String').SetString( 'library
13514:   ProjectTypeProgram','String').SetString( 'program
13515:   Personality32Bit','String').SetString( '32 bit
13516:   Personality64Bit','String').SetString( '64 bit
13517:   PersonalityDelphi','String').SetString( 'Delphi
13518:   PersonalityDelphiDotNet','String').SetString( 'Delphi.net
13519:   PersonalityBCB','String').SetString( 'C++Builder
13520:   PersonalityCSB','String').SetString( 'C#Builder
13521:   PersonalityVB','String').SetString( 'Visual Basic
13522:   PersonalityDesign','String').SetString( 'Design
13523:   PersonalityUnknown','String').SetString( 'Unknown personality
13524:   PersonalityBDS','String').SetString( 'Borland Developer Studio
13525:   DOFDirectoriesSection','String').SetString( 'Directories
13526:   DOFUnitOutputDirKey','String').SetString( 'UnitOutputDir
13527:   DOFSearchPathName','String').SetString( 'SearchPath
13528:   DOFConditionals','String').SetString( 'Conditionals
13529:   DOFLinkerSection','String').SetString( 'Linker
```

```

13530: DOFPackagesKey','String').SetString( 'Packages
13531: DOFCompilerSection','String').SetString( 'Compiler
13532: DOFPackageNoLinkKey','String').SetString( 'PackageNoLink
13533: DOFAdditionalSection','String').SetString( 'Additional
13534: DOFOptionsKey','String').SetString( 'Options
13535: TJclBorPersonality', '( bpDelphi32, bpDelphi64, bpBCBuilder32, b'
13536:   + 'pBCBuilder64, bpDelphiNet32, bpDelphiNet64, bpCSBuilder32, bpCSBuilder64, '
13537:   + 'bpVisualBasic32, bpVisualBasic64, bpDesign, bpUnknown )
13538: TJclBorPersonalities', 'set of TJclBorPersonality
13539: TJclBorDesigner', '( bdVCL, bdCLX )
13540: TJclBorDesigners', 'set of TJclBorDesigner
13541: TJclBorPlatform', '( bp32bit, bp64bit )
13542: FindClass('TOBJECT'),'TJclBorRADToolInstallation
13543: SIRegister_TJclBorRADToolInstallationObject(CL);
13544: SIRegister_TJclBorlandOpenHelp(CL);
13545: TJclHelp2Object', '( hoRegisterSession, hoRegister, hoPlugin )
13546: TJclHelp2Objects', 'set of TJclHelp2Object
13547: SIRegister_TJclHelp2Manager(CL);
13548: SIRegister_TJclBorRADToolIdeTool(CL);
13549: SIRegister_TJclBorRADToolIdePackages(CL);
13550: SIRegister_IJclCommandLineTool(CL);
13551: FindClass('TOBJECT'),'EJclCommandLineToolError
13552: SIRegister_TJclCommandLineTool(CL);
13553: SIRegister_TJclBorlandCommandLineTool(CL);
13554: SIRegister_TJclBCC32(CL);
13555: SIRegister_TJclDCC32(CL);
13556: TJclDCC', 'TJclDCC32
13557: SIRegister_TJclBpr2Mak(CL);
13558: SIRegister_TJclBorlandMake(CL);
13559: SIRegister_TJclBorRADToolPalette(CL);
13560: SIRegister_TJclBorRADToolRepository(CL);
13561: TCommandLineTool', '( clAsm, clBcc32, clDcc32, clDccIL, clMake, clProj2Mak )
13562: TCommandLineTools', 'set of TCommandLineTool
13563: //TJclBorRADToolInstallationClass', 'class of TJclBorRADToolInstallation
13564: SIRegister_TJclBorRADToolInstallation(CL);
13565: SIRegister_TJclBCBInstallation(CL);
13566: SIRegister_TJclDelphiInstallation(CL);
13567: SIRegister_TJclDCCIL(CL);
13568: SIRegister_TJclBDSInstallation(CL);
13569: TTraverseMethod', 'Function ( Installation : TJclBorRADToolInstallation) : Boolean
13570: SIRegister_TJclBorRADToolInstallations(CL);
13571: Function BPLFileName( const BPLPath, PackageFileName : string) : string
13572: Function BinaryFileName( const OutputPath, ProjectFileName : string) : string
13573: Function IsDelphiPackage( const FileName : string) : Boolean
13574: Function IsDelphiProject( const FileName : string) : Boolean
13575: Function IsBCBPackage( const FileName : string) : Boolean
13576: Function IsBCBProject( const FileName : string) : Boolean
13577: Procedure GetDPRFileInfo(const DPRFileName:string;out BinaryExtensio:string;const LibSuffix:PString);
13578: Procedure GetBPRFileInfo(const BPRFileName:string;out BinaryFileName:string;const Descript:PString);
13579: Procedure GetDPKFileInfo(const DPKFileName:string;out RunOnly:Bool;const LibSuffix:PString;const
Descript:PString;
13580: Procedure GetBPKFileInfo(const BPKFileName:string;out RunOnly:Bool;const BinaryFNme:PString;const
Descript:PString
13581: function SamePath(const Path1, Path2: string): Boolean;
13582: end;
13583:
13584: procedure SIRegister_JclFileUtils_max(CL: TPSPascalCompiler);
13585: begin
13586:   'ERROR_NO_MORE_FILES','LongInt').SetInt( 18);
13587:   //Function stat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13588:   //Function fstat64( FileDes: Integer;var StatBuffer : TStatBuf64) : Integer
13589:   //Function lstat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13590:   'LPathSeparator','String').SetString( '/'
13591:   'LDirDelimiter','String').SetString( '/'
13592:   'LDirSeparator','String').SetString( ':'
13593:   'JXPathDevicePrefix','String').SetString( '\\.\
13594:   'JXPathSeparator','String').SetString( '\
13595:   'JXDirDelimiter','String').SetString( '\
13596:   'JXDirSeparator','String').SetString( ';'
13597:   'JXPathUncPrefix','String').SetString( '\\
13598:   'faNormalFile','LongWord').SetUInt( $00000080);
13599:   //faUnixSpecific','').SetString( faSymLink);
13600:   JXTCompactPath', '( cpCenter, cpEnd )
13601:   _WIN32_FILE_ATTRIBUTE_DATA', 'record dwFileAttributes : DWORD; f'
13602:   + 'tCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime : '
13603:   + 'TFileTime; nFileSizeHigh : DWORD; nFileSizeLow : DWORD; end
13604:   TWin32FileAttributeData', '_WIN32_FILE_ATTRIBUTE_DATA
13605:   WIN32_FILE_ATTRIBUTE_DATA', '_WIN32_FILE_ATTRIBUTE_DATA
13606:
13607: Function jxPathAddSeparator( const Path : string) : string
13608: Function jxPathAddExtension( const Path, Extension : string) : string
13609: Function jxPathAppend( const Path, Append : string) : string
13610: Function jxPathBuildRoot( const Drive : Byte) : string
13611: Function jxPathCanonicalize( const Path : string) : string
13612: Function jxPathCommonPrefix( const Path1, Path2 : string) : Integer
13613: Function jxPathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string
13614: Procedure jxPathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
13615: Function jxPathExtractFileDirFixed( const S : string) : string
13616: Function jxPathExtractFileNameNoExt( const Path : string) : string

```

```

13617: Function jxPathExtractPathDepth( const Path : string; Depth : Integer ) : string
13618: Function jxPathGetDepth( const Path : string ) : Integer
13619: Function jxPathGetLongName( const Path : string ) : string
13620: Function jxPathGetShortName( const Path : string ) : string
13621: Function jxPathGetLongName( const Path : string ) : string
13622: Function jxPathGetShortName( const Path : string ) : string
13623: Function jxPathGetRelativePath( Origin, Destination : string ) : string
13624: Function jxPathGetTempPath : string
13625: Function jxPathIsAbsolute( const Path : string ) : Boolean
13626: Function jxPathIsChild( const Path, Base : string ) : Boolean
13627: Function jxPathIsDiskDevice( const Path : string ) : Boolean
13628: Function jxPathIsUNC( const Path : string ) : Boolean
13629: Function jxPathRemoveSeparator( const Path : string ) : string
13630: Function jxPathRemoveExtension( const Path : string ) : string
13631: Function jxPathGetPhysicalPath( const LocalizedPath : string ) : string
13632: Function jxPathGetLocalizedPath( const PhysicalPath : string ) : string
13633: JxTFileListOption', '( flFullNames, flRecursive, flMaskedSubfolders)
13634: JxTFileListOptions', 'set of TFileListOption
13635: JxTJclAttributeMatch', '( amAny, amExact, amSubSetOf, amSuperSetOf, amCustom )
13636: TFileHandler', 'Procedure ( const FileName : string)
13637: TFileHandlerEx', 'Procedure ( const Directory : string; const FileInfo : TSearchRec)
13638: Function BuildFileList( const Path : string; const Attr : Integer; const List : TStrings ) : Boolean
13639: //Function AdvBuildFileList( const Path : string; const Attr : Integer; const Files : TStrings; const
AttributeMatch:TJclAttributeMatch;const Optis:TFileListOptions;const SubfoldersMask:string;const
FileMatchFunc:TFileMatchFunc):Bool;
13640: Function jxVerifyFileAttributeMask( var RejectedAttributes, RequiredAttributes : Int ) : Boolean
13641: Function jxIsFileAttributeMatch( FileAttributes, RejectedAttributes, RequiredAttributes: Int ): Boolean;
13642: Function jxFileAttributesStr( const FileInfo : TSearchRec ) : string
13643: Function jxIsFileNameMatch( FileName: string; const Mask: string; const CaseSensitive: Boolean ): Boolean;
13644: Procedure jxEnumFiles( const Path: string; HandleFile: TFileHandlerEx;
RejectedAttributes: Integer; RequiredAttributes : Integer; Abort : TObjcet )
13645: Procedure jxEnumDirectories( const Root: string; const HandleDirectory: TFileHandler; const
IncludeHiddenDirects: Boolean; const SubDirectoriesMask: string; Abort: TObjcet; ResolveSymLinks: Bool )
13646: Procedure jxCreateEmptyFile( const FileName : string )
13647: Function jxCloseVolume( var Volume : THandle ) : Boolean
13648: Function jxDeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean ) : Boolean
13649: Function jxCopyDirectory( ExistingDirectoryName, NewDirectoryName : string ) : Boolean
13650: Function jxMoveDirectory( ExistingDirectoryName, NewDirectoryName : string ) : Boolean
13651: Function jxDelTree( const Path : string ) : Boolean
13652: //Function DelTreeEx(const Path:string;AbortOnFailure:Boolean; Progress:TDelTreeProgress):Boolean
13653: Function jxDiskInDrive( Drive : Char ) : Boolean
13654: Function jxDirectoryExists( const Name : string; ResolveSymLinks : Boolean ) : Boolean
13655: Function jxFileCreateTemp( var Prefix : string ) : THandle
13656: Function jxFileBackup( const FileName : string; Move : Boolean ) : Boolean
13657: Function jxFileCopy( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean ) : Boolean
13658: Function jxFileDelete( const FileName : string; MoveToRecycleBin : Boolean ) : Boolean
13659: Function jxFileExists( const FileName : string ) : Boolean
13660: Function jxFileMove( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean ) : Boolean
13661: Function jxFileRestore( const FileName : string ) : Boolean
13662: Function jxGetBackupFileName( const FileName : string ) : string
13663: Function jxIsBackupFileName( const FileName : string ) : Boolean
13664: Function jxFileGetDisplayName( const FileName : string ) : string
13665: Function jxFileGetGroupName( const FileName : string; ResolveSymLinks : Boolean ) : string
13666: Function jxFileGetOwnerName( const FileName : string; ResolveSymLinks : Boolean ) : string
13667: Function jxFileGetSize( const FileName : string ) : Int64
13668: Function jxFileGetTempName( const Prefix : string ) : string
13669: Function jxFileGetTypeNames( const FileName : string ) : string
13670: Function jxFindUnusedFileName( FileName: string; const FileExt : string; NumberPrefix : string ) : string
13671: Function jxForceDirectories( Name : string ) : Boolean
13672: Function jxGetDirectorySize( const Path : string ) : Int64
13673: Function jxGetDriveTypeStr( const Drive : Char ) : string
13674: Function jxGetFileAgeCoherence( const FileName : string ) : Boolean
13675: Procedure jxGetFileAttributeList( const Items : TStrings; const Attr : Integer )
13676: Procedure jxGetFileAttributeListEx( const Items : TStrings; const Attr : Integer )
13677: Function jxGetFileInformation( const FileName : string; out FileInfo : TSearchRec ) : Boolean;
13678: Function jxGetFileInformation1( const FileName : string ) : TSearchRec;
13679: //Function GetFileStatus(const FileName:string;out StatBuf:TStatBuf64;const
ResolveSymLinks:Boolean):Integer
13680: Function jxGetFileLastWrite( const FName : string ) : TFileTime;
13681: Function jxGetFileLastWrite1( const FName : string; out LocalTime : TDateTime ) : Boolean;
13682: Function jxGetFileLastAccess( const FName : string ) : TFileTime;
13683: Function jxGetFileLastAccess1( const FName : string; out LocalTime : TDateTime ) : Boolean;
13684: Function jxGetFileCreation( const FName : string ) : TFileTime;
13685: Function jxGetFileCreation1( const FName : string; out LocalTime : TDateTime ) : Boolean;
13686: Function jxGetFileLastWrite( const FName : string; out TimeStamp: Integer; ResolveSymLinks : Bool ): Bool;
13687: Function jxGetFileLastWrite1( const FName: string; out LocalTime: TDateTime; ResolveSymLinks: Bool ): Bool;
13688: Function jxGetFileLastWrite2( const FName : string; ResolveSymLinks : Boolean ) : Integer;
13689: Function jxGetFileLastAccess( const FName: string; out TimeStamp: Integer; ResolveSymLinks: Bool ): Bool;
13690: Function jxGetFileLastAccess1( const FName: string; out LocalTime: TDateTime; ResolveSymLinks: Bool ): Bool;
13691: Function jxGetFileLastAccess2( const FName: string; ResolveSymLinks: Boolean ): Integer;
13692: Function jxGetFileLastAttrChange( const FName: string; out TimeStamp: Integer; ResolveSymLinks: Bool ): Bool;
13693: Function jxGetFileLastAttrChange1( const FName: string; out LocalTime: TDateTime; ResolveSymLinks: Bool ): Bool;
13694: Function jxGetFileLastAttrChange2( const FName : string; ResolveSymLinks: Boolean ): Integer;
13695: Function jxGetModulePath( const Module : HMODULE ) : string
13696: Function jxGetSizeOfFile( const FileName : string ) : Int64;
13697: Function jxGetSizeOfFile1( const FileInfo : TSearchRec ) : Int64;
13698: Function jxGetSizeOfFile2( Handle : THandle ) : Int64;
13699: Function jxGetStandardFileInfo( const FileName : string ) : TWin32FileAttributeData
13700: Function jxIsDirectory( const FileName : string; ResolveSymLinks : Boolean ) : Boolean

```

```

13701: Function jxIsRootDirectory( const CanonicFileName : string) : Boolean
13702: Function jxLockVolume( const Volume : string; var Handle : THandle) : Boolean
13703: Function jxOpenVolume( const Drive : Char) : THandle
13704: Function jxSetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
13705: Function jxSetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
13706: Function jxSetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
13707: Function jxSetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
13708: Function jxSetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
13709: Function jxSetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
13710: Procedure jxShredFile( const FileName : string; Times : Integer)
13711: Function jxUnlockVolume( var Handle : THandle) : Boolean
13712: Function jxCreateSymbolicLink( const Name, Target : string) : Boolean
13713: Function jxSymbolicLinkTarget( const Name : string) : string
13714: TAttributeInterest', '( aiIgnored, aiRejected, aiRequired )
13715: SIRegister_TJclCustomFileAttrMask(CL);
13716: SIRegister_TJclFileAttributeMask(CL);
13717: TFileSearchOption', '( fsIncludeSubDirectories, fsIncludeHiddenS'
13718: +ubDirectories, fsLastChangeAfter, fsLastChangeBefore, fsMaxSize, fsMinSize)
13719: TFileSearchOptions', 'set of TFileSearchOption
13720: TFileSearchTaskID', 'Integer
13721: TFileSearchTerminationEvent', 'Procedure ( const ID : TFileSearch'
13722: +hTaskID; const Aborted : Boolean)
13723: TFileEnumeratorSyncMode', '( smPerFile, smPerDirectory )
13724: SIRegister_IJclFileEnumerator(CL);
13725: SIRegister_TJclFileEnumerator(CL);
13726: Function JxFileSearch : IJclFileEnumerator
13727: JxTFileFlag', '( ffDebug, ffInfoInferred, ffPatched, ffPreRelease, ffPrivateBuild, ffSpecialBuild )
13728: JxTFileFlags', 'set of TFileFlag
13729: FindClass('TOBJECT'), 'EJclFileVersionInfoError
13730: SIRegister_TJclFileVersionInfo(CL);
13731: Function jxOSIdentToString( const OSIdent : DWORD) : string
13732: Function jxOSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
13733: Function jxVersionResourceAvailable( const FileName : string) : Boolean
13734: TFileVersionFormat', '( vfMajorMinor, vfFull )
13735: Function jxFormatVersionString( const HiV, LoV : Word) : string;
13736: Function jxFormatVersionString1( const Major, Minor, Build, Revision : Word) : string;
13737: //Function FormatVersionString2( const FixedInfo : TVSFixedFileInfo; VersionFormat: TFileVersionFormat):str;
13738: //Procedure VersionExtractFileInfo( const FixedInfo: TVSFixedFileInfo; var Major, Minor, Build, Revision: Word);
13739: //Procedure VersionExtractProductInfo( const FixedInfo: TVSFixedFileInfo; var Major, Minor, Build,
Revision: Word);
13740: //Function VersionFixedFileInfo( const FileName : string; var FixedInfo : TVSFixedFileInfo) : Boolean
13741: Function jxVersionFixedFileInfoString( const FileName : string; VersionFormat : TFileVersionFormat; const
NotAvailableText : string) : string
13742: SIRegister_TJclTempFileStream(CL);
13743: FindClass('TOBJECT'), 'TJclCustomFileMapping
13744: SIRegister_TJclFileMappingView(CL);
13745: TJclFileMappingRoundOffset', '( rvDown, rvUp )
13746: SIRegister_TJclCustomFileMapping(CL);
13747: SIRegister_TJclFileMapping(CL);
13748: SIRegister_TJclSwapFileMapping(CL);
13749: SIRegister_TJclFileMappingStream(CL);
13750: TJclMappedTextReaderIndex', '( tiNoIndex, tiFull )
13751: //PPCharArray', '^TPCharArray // will not work
13752: SIRegister_TJclMappedTextReader(CL);
13753: SIRegister_TJclFileMaskComparator(CL);
13754: FindClass('TOBJECT'), 'EJclPathError
13755: FindClass('TOBJECT'), 'EJclFileUtilsError
13756: FindClass('TOBJECT'), 'EJclTempFileStreamError
13757: FindClass('TOBJECT'), 'EJclTempFileStreamError
13758: FindClass('TOBJECT'), 'EJclFileMappingError
13759: FindClass('TOBJECT'), 'EJclFileMappingViewError
13760: Function jxPathGetLongName2( const Path : string) : string
13761: Function jxWin32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13762: Function jxWin32MoveFileReplaceExisting( const SrcFileName, DstFileName : string) : Boolean
13763: Function jxWin32BackupFile( const FileName : string; Move : Boolean) : Boolean
13764: Function jxWin32RestoreFile( const FileName : string) : Boolean
13765: Function jxSamePath( const Path1, Path2 : string) : Boolean
13766: Procedure jxPathListAddItems( var List : string; const Items : string)
13767: Procedure jxPathListIncludeItems( var List : string; const Items : string)
13768: Procedure jxPathListDelItems( var List : string; const Items : string)
13769: Procedure jxPathListDelItem( var List : string; const Index : Integer)
13770: Function jxPathListItemCount( const List : string) : Integer
13771: Function jxPathListItemGetItem( const List : string; const Index : Integer) : string
13772: Procedure jxPathListItemSetItem( var List : string; const Index : Integer; const Value : string)
13773: Function jxPathListItemIndex( const List, Item : string) : Integer
13774: Function jxParamName(Idx: Int; const Separator: string; const AllowedPrefixChars: string; TrimName: Bool): string;
13775: Function jxParamValue(Index : Integer; const Separator : string; TrimValue : Boolean) : string;
13776: Function jxParamValue1( const SearchName: string; const Separator : string; CaseSensitive : Boolean; const
AllowedPrefixCharacters : string; TrimValue : Boolean) : string;
13777: Function jxParamPos( const SearchName : string; const Separator : string; CaseSensitive : Boolean; const
AllowedPrefixCharacters : string) : Integer
13778: end;
13779:
13780: procedure SIRegister_FileUtil(CL: TPSPascalCompiler);
13781: begin
13782: 'UTF8FileHeader', 'String').SetString( #Sef#Sbb#Sbf);
13783: Function lCompareFileNames( const Filename1, Filename2 : string) : integer
13784: Function lCompareFileNamesIgnoreCase( const Filename1, Filename2 : string) : integer
13785: Function lCompareFileNames( const Filename1, Filename2 : string; ResolveLinks : boolean) : integer

```



```

13786: Function lCompareFileNames(Filename1:PChar;Len1:int;Filename2:PChar;Len2:int;ResolveLinks:boolean):int;
13787: Function lFilenameIsAbsolute( const TheFilename : string) : boolean
13788: Function lFilenameIsWinAbsolute( const TheFilename : string) : boolean
13789: Function lFilenameIsUnixAbsolute( const TheFilename : string) : boolean
13790: Procedure lCheckIfFileIsExecutable( const AFilename : string)
13791: Procedure lCheckIfFileIsSymlink( const AFilename : string)
13792: Function lFileIsReadable( const AFilename : string) : boolean
13793: Function lFileIsWritable( const AFilename : string) : boolean
13794: Function lFileIsText( const AFilename : string) : boolean
13795: Function lFileIsText( const AFilename : string; out FileReadable : boolean) : boolean
13796: Function lFileIsExecutable( const AFilename : string) : boolean
13797: Function lFileIsSymlink( const AFilename : string) : boolean
13798: Function lFileIsHardLink( const AFilename : string) : boolean
13799: Function lFileSize( const Filename : string) : int64;
13800: Function lGetFileDescription( const AFilename : string) : string
13801: Function lReadAllLinks( const Filename : string; ExceptionOnError : boolean) : string
13802: Function lTryReadAllLinks( const Filename : string) : string
13803: Function lDirPathExists( const FileName : String) : Boolean
13804: Function lForceDirectory( DirectoryName : string) : boolean
13805: Function lDeleteDirectory( const DirectoryName : string; OnlyChildren : boolean) : boolean
13806: Function lProgramDirectory : string
13807: Function lDirectoryIsWritable( const DirectoryName : string) : boolean
13808: Function lExtractFileNameOnly( const AFilename : string) : string
13809: Function lExtractFileNameWithoutExt( const AFilename : string) : string
13810: Function lCompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
13811: Function lCompareFileExt( const Filename, Ext : string) : integer;
13812: Function lFilenameIsPascalUnit( const Filename : string) : boolean
13813: Function lAppendPathDelim( const Path : string) : string
13814: Function lChompPathDelim( const Path : string) : string
13815: Function lTrimFilename( const AFilename : string) : string
13816: Function lCleanAndExpandFilename( const Filename : string) : string
13817: Function lCleanAndExpandDirectory( const Filename : string) : string
13818: Function lCreateAbsolutePath( const SearchPath, BaseDirectory : string) : string
13819: Function lCreateRelativePath( const Filename, BaseDirectory : string; UsePointDirectory : boolean;
AlwaysRequireSharedBaseFolder : Boolean) : string
13820: Function lCreateAbsolutePath( const Filename, BaseDirectory : string) : string
13821: Function lFileIsInPath( const Filename, Path : string) : boolean
13822: Function lFileIsInDirectory( const Filename, Directory : string) : boolean
13823: TSearchFileInPathFlag', '( sffDontSearchInBasePath, sffSearchLoUpCase )
13824: TSearchFileInPathFlags', 'set of TSearchFileInPathFlag
13825: 'AllDirectoryEntriesMask', 'String').SetString( '*'
13826: Function lGetAllFilesMask : string
13827: Function lGetExeExt : string
13828: Function lSearchFileInPath( const Filename, BasePath, SearchPath, Delimiter : string; Flags :
TSearchFileInPathFlags) : string
13829: Function lSearchAllFilesInPath( const Filename, BasePath, SearchPath, Delimiter:string;Flags :
TSearchFileInPathFlags) : TStringList
13830: Function lFindDiskFilename( const Filename : string) : string
13831: Function lFindDiskFileCaseInsensitive( const Filename : string) : string
13832: Function lFindDefaultExecutablePath( const Executable : string; const BaseDir: string):string
13833: Function lGetDarwinSystemFilename( Filename : string) : string
13834: SIRegister_TFileIterator(CL);
13835: TFileFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13836: TDirectoryFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13837: TDirectoryEnterEvent', 'Procedure ( FileIterator : TFileIterator)
13838: SIRegister_TFileSearcher(CL);
13839: Function lFindAllFiles(const SearchPath:String;SearchMsk:String;SearchSubDirs:Bool):TStringList
13840: Function lFindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
13841: // TCopyFileFlag', '( cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime )
13842: // TCopyFileFlags', 'set of TCopyFileFlag
13843: Function lCopyFile( const SrcFilename, DestFilename : string; Flags : TCopyFileFlags) : boolean
13844: Function lCopyFile( const SrcFilename, DestFilename : string; PreserveTime : boolean) : boolean
13845: Function lCopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
13846: Function lReadFileToString( const Filename : string) : string
13847: Function lGetTempFilename( const Directory, Prefix : string) : string
13848: {Function NeedRTLAnsi : boolean
13849: Procedure SetNeedRTLAnsi( NewValue : boolean)
13850: Function UTF8ToSys( const s : string) : string
13851: Function SysToUTF8( const s : string) : string
13852: Function ConsoleToUTF8( const s : string) : string
13853: Function UTF8ToConsole( const s : string) : string}
13854: Function FileExistsUTF8( const Filename : string) : boolean
13855: Function FileAgeUTF8( const FileName : string) : Longint
13856: Function DirectoryExistsUTF8( const Directory : string) : Boolean
13857: Function ExpandFileNameUTF8( const FileName : string) : string
13858: Function ExpandUNCFileNameUTF8( const FileName : string) : string
13859: Function ExtractShortPathNameUTF8( const FileName : String) : String
13860: Function FindFirstUTF8(const Path: string; Attr : Longint; out Rslt : TSearchRec) : Longint
13861: Function FindNextUTF8( var Rslt : TSearchRec) : Longint
13862: Procedure FindCloseUTF8( var F : TSearchRec)
13863: Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
13864: Function FileGetAttrUTF8( const FileName : String) : Longint
13865: Function FileSetAttrUTF8( const FileName : String; Attr : longint) : Longint
13866: Function DeleteFileUTF8( const FileName : String) : Boolean
13867: Function RenameFileUTF8( const OldName, NewName : String) : Boolean
13868: Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
13869: Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
13870: Function GetCurrentDirUTF8 : String
13871: Function SetCurrentDirUTF8( const NewDir : String) : Boolean

```

```

13872: Function CreateDirUTF8( const NewDir : String) : Boolean
13873: Function RemoveDirUTF8( const Dir : String) : Boolean
13874: Function ForceDirectoriesUTF8( const Dir : string) : Boolean
13875: Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
13876: Function FileCreateUTF8( const FileName : string) : THandle;
13877: Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
13878: Function ParamStrUTF8( Param : Integer) : string
13879: Function GetEnvironmentStringUTF8( Index : Integer) : string
13880: Function GetEnvironmentVariableUTF8( const EnvVar : string) : String
13881: Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
13882: Function GetAppConfigFileUTF8(Global: Boolean; SubDir: boolean; CreateDir : boolean) : string
13883: Function SysErrorMessageUTF8( ErrorCode : Integer) : String
13884: end;
13885:
13886: procedure SIRegister_Keyboard(CL: TPSPascalCompiler);
13887: begin
13888:     //VK_F23 = 134;
13889:     //{$EXTERNALSYM VK_F24}
13890:     //VK_F24 = 135;
13891:     TVirtualKeyCode', 'Integer
13892:     'VK_MOUSEWHEELUP', 'integer').SetInt(134);
13893:     'VK_MOUSEWHEELDOWN', 'integer').SetInt(135);
13894: Function glIsKeyDown( c : Char) : Boolean;
13895: Function glIsKeyDown1( vk : TVirtualKeyCode) : Boolean;
13896: Function glKeyPressed( minVkCode : TVirtualKeyCode) : TVirtualKeyCode
13897: Function glVirtualKeyCodeToKeyName( vk : TVirtualKeyCode) : String
13898: Function glKeyNameToVirtualKeyCode( const keyName : String) : TVirtualKeyCode
13899: Function glCharToVirtualKeyCode( c : Char) : TVirtualKeyCode
13900: Procedure glKeyboardNotifyWheelMoved( wheelDelta : Integer)
13901: end;
13902:
13903: procedure SIRegister_GLCrossPlatform(CL: TPSPascalCompiler);
13904: begin
13905:     TGLPoint', 'TPoint
13906:     //PGLPoint', '^TGLPoint // will not work
13907:     TGLRect', 'TRect
13908:     //PGLRect', '^TGLRect // will not work
13909:     TDelphiColor', 'TColor
13910:     TGLPicture', 'TPicture
13911:     TGLGraphic', 'TGraphic
13912:     TGLBitmap', 'TBitmap
13913:     //TGraphicClass', 'class of TGraphic
13914:     TGLTextLayout', '( tlTop, tlCenter, tlBottom )
13915:     TGLMouseButton', '( mbLeft, mbRight, mbMiddle )
13916:     TGLMouseEvent', 'Procedure ( Sender : TObject; Button : TGLMouse'
13917:     + 'Button; Shift : TShiftState; X, Y : Integer)
13918:     TGLMouseMoveEvent', 'TMouseMoveEvent
13919:     TGLKeyEvent', 'TKeyEvent
13920:     TGLKeyPressEvent', 'TKeyPressEvent
13921:     EGLOSError', 'EWin32Error
13922:     EGLOSError', 'EWin32Error
13923:     EGLOSError', 'EOSError
13924:     'glsAllFilter', 'string').SetString('All // sAllFilter
13925: Function GLPoint( const x, y : Integer) : TGLPoint
13926: Function GLRGB( const r, g, b : Byte) : TColor
13927: Function GLColorToRGB( color : TColor) : TColor
13928: Function GLGetRValue( rgb : DWORD) : Byte
13929: Function GLGetGValue( rgb : DWORD) : Byte
13930: Function GLGetBValue( rgb : DWORD) : Byte
13931: Procedure GLInitWinColors
13932: Function GLRect( const aLeft, aTop, aRight, aBottom : Integer) : TGLRect
13933: Procedure GLInflateGLRect( var aRect : TGLRect; dx, dy : Integer)
13934: Procedure GLIntersectGLRect( var aRect : TGLRect; const rect2 : TGLRect)
13935: Procedure GLInformationDlg( const msg : String)
13936: Function GLQuestionDlg( const msg : String) : Boolean
13937: Function GLInputDialog( const aCaption, aPrompt, aDefault : String) : String
13938: Function GLSavePictureDialog( var aFileName : String; const aTitle : String) : Boolean
13939: Function GLOpenPictureDialog( var aFileName : String; const aTitle : String) : Boolean
13940: Function GLApplicationTerminated : Boolean
13941: Procedure GLRaiseLastOSError
13942: Procedure GLFreeAndNil( var anObject: TObject)
13943: Function GLGetDeviceLogicalPixelsX( device : Cardinal) : Integer
13944: Function GLGetCurrentColorDepth : Integer
13945: Function GLPixelFormatToColorBits( aPixelFormat : TPixelFormat) : Integer
13946: Function GLBitmapScanLine( aBitmap : TGLBitmap; aRow : Integer) : Pointer
13947: Procedure GLSleep( length : Cardinal)
13948: Procedure GLQueryPerformanceCounter( var val : Int64)
13949: Function GLQueryPerformanceFrequency( var val : Int64) : Boolean
13950: Function GLStartPrecisionTimer : Int64
13951: Function GLPrecisionTimerLap( const precisionTimer : Int64) : Double
13952: Function GLStopPrecisionTimer( const precisionTimer : Int64) : Double
13953: Function GLRDTSC : Int64
13954: Procedure GLLoadBitmapFromInstance( ABitmap : TBitmap; AName : string)
13955: Function GLOKMessageBox( const Text, Caption : string) : Integer
13956: Procedure GLShowHTMLUrl( Url : String)
13957: Procedure GLShowCursor( AShow : boolean)
13958: Procedure GLSetCursorPos( AScreenX, AScreenY : integer)
13959: Procedure GLGetCursorPos( var point : TGLPoint)
13960: Function GLGetScreenWidth : integer

```

```

13961: Function GLGetScreenHeight : integer
13962: Function GLGetTickCount : int64
13963: function RemoveSpaces(const str : String) : String;
13964: Function TNormalMapSpace', '( nmsObject, nmsTangent )
13965: Procedure CalcObjectSpaceLightVectors(Light:TAffineVector;Vertices TAffineVectorList;Colors:TVectorList)
13966: Procedure SetupTangentSpace( Vertices, Normals, TexCoords, Tangents, BiNormals : TAffineVectorList)
13967: Procedure CalcTangentSpaceLightVectors( Light : TAffineVector; Vertices, Normals, Tangents, BiNormals :
    TAffineVectorList; Colors : TVectorList)
13968: Function CreateObjectSpaceNormalMap( Width,Height:Integer;HiNormals,
    HiTexCoords:TAffineVectorList):TGLBitmap
13969: Function CreateTangentSpaceNormalMap( Width, Height : Integer; HiNormals, HiTexCoords, LoNormals,
    LoTexCoords, Tangents, BiNormals : TAffineVectorList ) : TGLBitmap
13970: end;
13971:
13972: procedure SIRegister_GLStarRecord(CL: TPSPascalCompiler);
13973: begin
13974:   TGLStarRecord', 'record RA: Word; DEC: SmallInt; BVColorIndex: Byte; VMagnitude: Byte; end
13975:   // PGLStarRecord', 'TGLStarRecord // will not work
13976: Function StarRecordPositionZUp( const starRecord : TGLStarRecord) : TAffineVector
13977: Function StarRecordPositionYUp( const starRecord : TGLStarRecord) : TAffineVector
13978: Function StarRecordColor( const starRecord : TGLStarRecord; bias : Single) : TVector
13979: end;
13980:
13981:
13982: procedure SIRegister_GeometryBB(CL: TPSPascalCompiler);
13983: begin
13984:   TAABB', 'record min : TAffineVector; max : TAffineVector; end
13985:   //PAABB', 'TAABB // will not work
13986:   TBSphere', 'record Center : TAffineVector; Radius : single; end
13987:   TClipRect', 'record Left : Single; Top:Single; Right:Single; Bottom : Single; end
13988:   TSpaceContains', '(scNoOverlap, scContainsFully, scContainsPartially )
13989: Function AddBB( var c1 : THmgBoundingBox; const c2 : THmgBoundingBox) : THmgBoundingBox
13990: Procedure AddAABB( var aabb : TAABB; const aabb1 : TAABB)
13991: Procedure SetBB( var c : THmgBoundingBox; const v : TVector)
13992: Procedure SetAABB( var bb : TAABB; const v : TVector)
13993: Procedure BBTransform( var c : THmgBoundingBox; const m : TMatrix)
13994: Procedure AABBBTransform( var bb : TAABB; const m : TMatrix)
13995: Procedure AABBScale( var bb : TAABB; const v : TAffineVector)
13996: Function BBMinX( const c : THmgBoundingBox) : Single
13997: Function BBMaxX( const c : THmgBoundingBox) : Single
13998: Function BBMinY( const c : THmgBoundingBox) : Single
13999: Function BBMaxY( const c : THmgBoundingBox) : Single
14000: Function BBMinZ( const c : THmgBoundingBox) : Single
14001: Function BBMaxZ( const c : THmgBoundingBox) : Single
14002: Procedure AABBIInclude( var bb : TAABB; const p : TAffineVector)
14003: Procedure AABBFFromSweep( var SweepAABB : TAABB; const Start, Dest : TVector; const Radius : Single)
14004: Function AABBIIntersection( const aabb1, aabb2 : TAABB) : TAABB
14005: Function BBToAABB( const aabb : THmgBoundingBox) : TAABB
14006: Function AABBTToBB( const anAABB : TAABB) : THmgBoundingBox;
14007: Function AABBTToBBI( const anAABB : TAABB; const m : TMatrix) : THmgBoundingBox;
14008: Procedure OffsetAABB( var aabb : TAABB; const delta : TAffineVector);
14009: Procedure OffsetAABB1( var aabb : TAABB; const delta : TVector);
14010: Function IntersectAABBs( const aabb1, aabb2 : TAABB; const m1To2, m2To1 : TMatrix) : Boolean;
14011: Function IntersectAABBsAbsoluteXY( const aabb1, aabb2 : TAABB) : Boolean
14012: Function IntersectAABBsAbsoluteXZ( const aabb1, aabb2 : TAABB) : Boolean
14013: Function IntersectAABBsAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14014: Function AABBFitsInAABBAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14015: Function PointInAABB( const p : TAffineVector; const aabb : TAABB) : Boolean;
14016: Function PointInAABB1( const p : TVector; const aabb : TAABB) : Boolean;
14017: Function PlaneIntersectAABB( Normal : TAffineVector; d : single; aabb : TAABB) : boolean
14018: Function TriangleIntersectAABB( const aabb : TAABB; v1, v2, v3 : TAffineVector) : boolean
14019: Procedure ExtractAABBCorners( const AABBB : TAABB; var AABBCorners : TAABBCorners)
14020: Procedure AABBTToBSphere( const AABBB : TAABB; var BSphere : TBSphere)
14021: Procedure BSphereToAABB( const BSphere : TBSphere; var AABBB : TAABB);
14022: Function BSphereToAABB1( const center : TAffineVector; radius : Single) : TAABB;
14023: Function BSphereToAABB2( const center : TVector; radius : Single) : TAABB;
14024: Function AABBBContainsAABB( const mainAABB, testAABB : TAABB) : TSpaceContains
14025: Function BSphereContainsAABB( const mainBSphere : TBSphere; const testAABB : TAABB) : TSpaceContains
14026: Function BSphereContainsBSphere( const mainBSphere, testBSphere : TBSphere) : TSpaceContains
14027: Function AABBBContainsBSphere( const mainAABB : TAABB; const testBSphere : TBSphere) : TSpaceContains
14028: Function PlaneContainsBSphere(const Location,Normal:TAffineVector;const
    testBSphere:TBSphere):TSpaceContains
14029: Function FrustumContainsBSphere( const Frustum : TFrustum; const testBSphere : TBSphere) : TSpaceContains
14030: Function FrustumContainsAABB( const Frustum : TFrustum; const testAABB : TAABB) : TSpaceContains
14031: Function ClipToAABB( const v : TAffineVector; const AABBB : TAABB) : TAffineVector
14032: Function BSphereIntersectsBSphere( const mainBSphere, testBSphere : TBSphere) : boolean
14033: Procedure IncludeInClipRect( var clipRect : TClipRect; x, y : Single)
14034: Function AABBTToClipRect(const aabb:TAABB;modelViewProjection:TMatrix;viewportSizeX,
    viewportSizeY:Int):TClipRect
14035: end;
14036:
14037: procedure SIRegister_GeometryCoordinates(CL: TPSPascalCompiler);
14038: begin
14039:   Procedure Cylindrical_Cartesian( const r, theta, z1 : single; var x, y, z : single);
14040:   Procedure Cylindrical_Cartesian1( const r, theta, z1 : double; var x, y, z : double);
14041:   Procedure Cylindrical_Cartesian2( const r,theta,z1 : single; var x, y, z : single; var ierr : integer);
14042:   Procedure Cylindrical_Cartesian3( const r,theta,z1 : double; var x, y, z : double; var ierr : integer);
14043:   Procedure Cartesian_Cylindrical( const x, y, z1 : single; var r, theta, z : single);
14044:   Procedure Cartesian_Cylindrical1( const x, y, z1 : double; var r, theta, z : double);

```

```

14045: Procedure Spherical_Cartesian( const r, theta, phi : single; var x, y, z : single);
14046: Procedure Spherical_Cartesian1( const r, theta, phi : double; var x, y, z : double);
14047: Procedure Spherical_Cartesian2( const r, theta, phi : single; var x, y, z : single; var ierr : integer);
14048: Procedure Spherical_Cartesian3( const r, theta, phi : double; var x, y, z : double; var ierr : integer);
14049: Procedure Cartesian_Spherical( const x, y, z : single; var r, theta, phi : single);
14050: Procedure Cartesian_Spherical1( const v : TAffineVector; var r, theta, phi : Single);
14051: Procedure Cartesian_Spherical2( const x, y, z : double; var r, theta, phi : double);
14052: Procedure ProlateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14053: Procedure ProlateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14054: Procedure ProlateSpheroidal_Cartesian2( const xi, eta, phi, a : single; var x, y, z : single; var ierr : integer);
14055: Procedure ProlateSpheroidal_Cartesian3( const xi, eta, phi, a : double; var x, y, z : double; var ierr : integer);
14056: Procedure OblateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14057: Procedure OblateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14058: Procedure OblateSpheroidal_Cartesian2( const xi, eta, phi, a : single; var x, y, z : single; var ierr : integer);
14059: Procedure OblateSpheroidal_Cartesian3( const xi, eta, phi, a : double; var x, y, z : double; var ierr : integer);
14060: Procedure BipolarCylindrical_Cartesian( const u, v, z1, a : single; var x, y, z : single);
14061: Procedure BipolarCylindrical_Cartesian1( const u, v, z1, a : double; var x, y, z : double);
14062: Procedure BipolarCylindrical_Cartesian2( const u, v, z1, a : single; var x, y, z : single; var ierr : integer);
14063: Procedure BipolarCylindrical_Cartesian3( const u, v, z1, a : double; var x, y, z : double; var ierr : integer);
14064: end;
14065:
14066: procedure SIRegister_VectorGeometry(CL: TPSPascalCompiler);
14067: begin
14068:   'EPSILON', 'Single').setExtended( 1e-40);
14069:   'EPSILON2', 'Single').setExtended( 1e-30); }
14070: TRenderContextClippingInfo', 'record origin : TVector; clippingD'
14071:   + 'irection : TVector; viewPortRadius : Single; farClippingDistance:Single;frustum:TFrustum; end
14072: THmgPlane', 'TVector
14073:   TDoubleHmgPlane', 'THomogeneousDblVector
14074: {TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
14075:   + 'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
14076:   + ', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW } )
14077: TSingleArray', 'array of Single
14078: TTransformations', 'array [0..15] of Single)
14079: TPackedRotationMatrix', 'array [0..2] of Smallint)
14080: TVertex', 'TAffineVector
14081: //TVectorGL', 'THomogeneousFltVector
14082: //TMatrixGL', 'THomogeneousFltMatrix
14083: // TPackedRotationMatrix = array [0..2] of SmallInt;
14084: Function glTexPointMake( const s, t : Single) : TTexPoint
14085: Function glAffineVectorMake( const x, y, z : Single) : TAffineVector;
14086: Function glAffineVectorMakel( const v : TVectorGL) : TAffineVector;
14087: Procedure glSetAffineVector( var v : TAffineVector; const x, y, z : Single);
14088: Procedure glSetVector( var v : TAffineVector; const x, y, z : Single);
14089: Procedure glSetVector1( var v : TAffineVector; const vSrc : TVectorGL);
14090: Procedure glSetVector2( var v : TAffineVector; const vSrc : TAffineVector);
14091: Procedure glSetVector3( var v : TAffineDblVector; const vSrc : TAffineVector);
14092: Procedure glSetVector4( var v : TAffineDblVector; const vSrc : TVectorGL);
14093: Function glVectorMake( const v : TAffineVector; w : Single) : TVectorGL;
14094: Function glVectorMakel( const x, y, z : Single; w : Single) : TVectorGL;
14095: Function glPointMake( const x, y, z : Single) : TVectorGL;
14096: Function glPointMakel( const v : TAffineVector) : TVectorGL;
14097: Function glPointMake2( const v : TVectorGL) : TVectorGL;
14098: Procedure glSetVector5( var v : TVectorGL; const x, y, z : Single; w : Single);
14099: Procedure glSetVector6( var v : TVectorGL; const av : TAffineVector; w : Single);
14100: Procedure glglSetVector7( var v : TVectorGL; const vSrc : TVectorGL);
14101: Procedure glMakePoint( var v : TVectorGL; const x, y, z : Single);
14102: Procedure glMakePoint1( var v : TVectorGL; const av : TAffineVector);
14103: Procedure glMakePoint2( var v : TVectorGL; const av : TVectorGL);
14104: Procedure glMakeVector( var v : TAffineVector; const x, y, z : Single);
14105: Procedure glMakeVector1( var v : TVectorGL; const x, y, z : Single);
14106: Procedure glMakeVector2( var v : TVectorGL; const av : TAffineVector);
14107: Procedure glMakeVector3( var v : TVectorGL; const av : TVectorGL);
14108: Procedure glRstVector( var v : TAffineVector);
14109: Procedure glRstVector1( var v : TVectorGL);
14110: Function glVectorAdd( const v1, v2 : TAffineVector) : TAffineVector;
14111: Procedure glVectorAdd1( const v1, v2 : TAffineVector; var vr : TAffineVector);
14112: //Procedure VectorAdd2( const v1, v2 : TAffineVector; vr : PAffineVector);
14113: Function glVectorAdd3( const v1, v2 : TVectorGL) : TVectorGL;
14114: Procedure glVectorAdd4( const v1, v2 : TVectorGL; var vr : TVectorGL);
14115: Function glVectorAdd5( const v : TAffineVector; const f : Single) : TAffineVector;
14116: Function glVectorAdd6( const v : TVectorGL; const f : Single) : TVectorGL;
14117: Procedure glAddVector7( var v1 : TAffineVector; const v2 : TAffineVector);
14118: Procedure glAddVector8( var v1 : TAffineVector; const v2 : TVectorGL);
14119: Procedure glAddVector9( var v1 : TVectorGL; const v2 : TVectorGL);
14120: Procedure glAddVector10( var v : TAffineVector; const f : Single);
14121: Procedure glAddVector11( var v : TVectorGL; const f : Single);
14122: //Procedure TexPointArrayAdd(const src:PTexPointArray;const delta:TTexPoint;const
nb:Int;dest:PTexPointArray);
14123: //Procedure TexPointArrayScaleAndAdd(const src:PTexPointArray;const delta:TTexPoint;const
nb:Integer;const scale: TTexPoint; dest : PTexPointArray);
14124: //Procedure VectorArrayAdd(const src:PAffineVectorArray;const delta:TAffineVector;const nb:Integer;dest:
PAffineVectorArray);
14125: Function glVectorSubtract( const V1, V2 : TAffineVector) : TAffineVector;
14126: Procedure glVectorSubtract1( const v1, v2 : TAffineVector; var result : TAffineVector);
14127: Procedure glVectorSubtract2( const v1, v2 : TAffineVector; var result : TVectorGL);
14128: Procedure glVectorSubtract3( const v1 : TVectorGL; v2 : TAffineVector; var result : TVectorGL);
14129: Function glVectorSubtract4( const V1, V2 : TVectorGL) : TVectorGL;
14130: Procedure glVectorSubtract5( const v1, v2 : TVectorGL; var result : TVectorGL);

```



```

14131: Procedure glVectorSubtract6( const v1, v2 : TVectorGL; var result : TAffineVector);
14132: Function glVectorSubtract7( const v1 : TAffineVector; delta : Single) : TAffineVector;
14133: Function glVectorSubtract8( const v1 : TVectorGL; delta : Single) : TVectorGL;
14134: Procedure glSubtractVector9( var V1 : TAffineVector; const V2 : TAffineVector);
14135: Procedure glSubtractVector10( var V1 : TVectorGL; const V2 : TVectorGL);
14136: Procedure glCombineVector( var vr : TAffineVector; const v : TAffineVector; var f : Single);
14137: //Procedure CombineVector1( var vr : TAffineVector; const v : TAffineVector; pf : PFloat);
14138: Function glTexPointCombine( const t1, t2 : TTexPoint; f1, f2 : Single) : TTexPoint
14139: Function glVectorCombine2( const V1, V2 : TAffineVector; const F1, F2 : Single) : TAffineVector;
14140: Function glVectorCombine33( const V1, V2, V3 : TAffineVector; const F1, F2, F3 : Single) : TAffineVector;
14141: Procedure glVectorCombine34( const V1,V2,V3:TAffineVector; const F1,F2,F3: Single; var vr : TAffineVector);
14142: Procedure glCombineVector5( var vr : TVectorGL; const v : TVectorGL; var f : Single);
14143: Procedure glCombineVector6( var vr : TVectorGL; const v : TAffineVector; var f : Single);
14144: Function glVectorCombine7( const V1, V2 : TVectorGL; const F1, F2 : Single) : TVectorGL;
14145: Function glVectorCombine8( const V1 : TVectorGL; const V2: TAffineVector; const F1,F2:Single) : TVectorGL;
14146: Procedure glVectorCombine9( const V1:TVectorGL; const V2:TAffineVector; const F1,F2:Single; var vr:TVectorGL);
14147: Procedure glVectorCombine10( const V1, V2 : TVectorGL; const F1, F2 : Single; var vr : TVectorGL);
14148: Procedure glVectorCombine11( const V1, V2 : TVectorGL; const F2 : Single; var vr : TVectorGL);
14149: Function glVectorCombine3( const V1, V2, V3 : TVectorGL; const F1, F2, F3 : Single) : TVectorGL;
14150: Procedure glVectorCombine31( const V1, V2, V3 : TVectorGL; const F1, F2, F3:Single; var vr : TVectorGL);
14151: Function glVectorDotProduct( const V1, V2 : TAffineVector) : Single;
14152: Function glVectorDotProduct1( const V1, V2 : TVectorGL) : Single;
14153: Function glVectorDotProduct2( const V1 : TVectorGL; const V2 : TAffineVector) : Single;
14154: Function glPointProject( const p, origin, direction : TAffineVector) : Single;
14155: Function glPointProject1( const p, origin, direction : TVectorGL) : Single;
14156: Function glVectorCrossProduct( const V1, V2 : TAffineVector) : TAffineVector;
14157: Function glVectorCrossProduct1( const V1, V2 : TVectorGL) : TVectorGL;
14158: Procedure glVectorCrossProduct2( const v1, v2 : TVectorGL; var vr : TVectorGL);
14159: Procedure glVectorCrossProduct3( const v1, v2 : TAffineVector; var vr : TVectorGL);
14160: Procedure glVectorCrossProduct4( const v1, v2 : TVectorGL; var vr : TAffineVector);
14161: Procedure glVectorCrossProduct5( const v1, v2 : TAffineVector; var vr : TAffineVector);
14162: Function glLerp( const start, stop, t : Single) : Single
14163: Function glAngleLerp( start, stop, t : Single) : Single
14164: Function glDistanceBetweenAngles( angle1, angle2 : Single) : Single
14165: Function glTexPointLerp( const t1, t2 : TTexPoint; t : Single) : TTexPoint;
14166: Function glVectorLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14167: Procedure glVectorLerp1( const v1, v2 : TAffineVector; t : Single; var vr : TAffineVector);
14168: Function glVectorLerp2( const v1, v2 : TVectorGL; t : Single) : TVectorGL;
14169: Procedure glVectorLerp3( const v1, v2 : TVectorGL; t : Single; var vr : TVectorGL);
14170: Function glVectorAngleLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14171: Function glVectorAngleCombine( const v1, v2 : TAffineVector; f : Single) : TAffineVector;
14172: // Procedure VectorArrayLerp( const src1, src2:PVectorArray; t:Single; n:Integer; dest:PVectorArray);
14173: // Procedure VectorArrayLerp1( const src1, src2 : PAffineVectorArray; t : Single; n : Integer; dest :
    PAffineVectorArray);
14174: Function glVectorLength( const x, y : Single) : Single;
14175: Function glVectorLength1( const x, y, z : Single) : Single;
14176: Function glVectorLength2( const v : TAffineVector) : Single;
14177: Function glVectorLength3( const v : TVectorGL) : Single;
14178: Function glVectorLength4( const v : array of Single) : Single;
14179: Function glVectorNorm( const x, y : Single) : Single;
14180: Function glVectorNorm1( const v : TAffineVector) : Single;
14181: Function glVectorNorm2( const v : TVectorGL) : Single;
14182: Function glVectorNorm3( var V : array of Single) : Single;
14183: Procedure glNormalizeVector( var v : TAffineVector);
14184: Procedure glNormalizeVector1( var v : TVectorGL);
14185: Function glVectorNormalize( const v : TAffineVector) : TAffineVector;
14186: Function glVectorNormalize1( const v : TVectorGL) : TVectorGL;
14187: // Procedure NormalizeVectorArray( list : PAffineVectorArray; n : Integer);
14188: Function glVectorAngleCosine( const V1, V2 : TAffineVector) : Single
14189: Function glVectorNegate( const v : TAffineVector) : TAffineVector;
14190: Function glVectorNegate1( const v : TVectorGL) : TVectorGL;
14191: Procedure glNegateVector( var V : TAffineVector);
14192: Procedure glNegateVector2( var V : TVectorGL);
14193: Procedure glNegateVector3( var V : array of Single);
14194: Procedure glScaleVector( var v : TAffineVector; factor : Single);
14195: Procedure glScaleVector1( var v : TAffineVector; const factor : TAffineVector);
14196: Procedure glScaleVector2( var v : TVectorGL; factor : Single);
14197: Procedure glScaleVector3( var v : TVectorGL; const factor : TVectorGL);
14198: Function glVectorScale( const v : TAffineVector; factor : Single) : TAffineVector;
14199: Procedure glVectorScale1( const v : TAffineVector; factor : Single; var vr : TAffineVector);
14200: Function glVectorScale2( const v : TVectorGL; factor : Single) : TVectorGL;
14201: Procedure glVectorScale3( const v : TVectorGL; factor : Single; var vr : TVectorGL);
14202: Procedure glVectorScale4( const v : TVectorGL; factor : Single; var vr : TAffineVector);
14203: Procedure glDivideVector( var v : TVectorGL; const divider : TVectorGL);
14204: Function glVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14205: Function glVectorEquals1( const V1, V2 : TAffineVector) : Boolean;
14206: Function glAffineVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14207: Function glVectorIsNull( const v : TVectorGL) : Boolean;
14208: Function glVectorIsNull1( const v : TAffineVector) : Boolean;
14209: Function glVectorSpacing( const v1, v2 : TTexPoint) : Single;
14210: Function glVectorSpacing1( const v1, v2 : TAffineVector) : Single;
14211: Function glVectorSpacing2( const v1, v2 : TVectorGL) : Single;
14212: Function glVectorDistance( const v1, v2 : TAffineVector) : Single;
14213: Function glVectorDistance1( const v1, v2 : TVectorGL) : Single;
14214: Function glVectorDistance2( const v1, v2 : TAffineVector) : Single;
14215: Function glVectorDistance21( const v1, v2 : TVectorGL) : Single;
14216: Function glVectorPerpendicular( const V, N : TAffineVector) : TAffineVector
14217: Function glVectorReflect( const V, N : TAffineVector) : TAffineVector
14218: Procedure glRotateVector( var vector : TVectorGL; const axis : TAffineVector; angle : Single);

```

```

14219: Procedure glRotateVector1( var vector : TVectorGL; const axis : TVectorGL; angle : Single);
14220: Procedure glRotateVectorAroundY( var v : TAffineVector; alpha : Single)
14221: Function glVectorRotateAroundX( const v : TAffineVector; alpha : Single) : TAffineVector;
14222: Function glVectorRotateAroundY( const v : TAffineVector; alpha : Single) : TAffineVector;
14223: Procedure glVectorRotateAroundZ( const v : TAffineVector; alpha : Single; var vr : TAffineVector);
14224: Function glVectorRotateAroundZ( const v : TAffineVector; alpha : Single) : TAffineVector;
14225: Procedure glAbsVector( var v : TVectorGL);
14226: Procedure glAbsVector1( var v : TAffineVector);
14227: Function glVectorAbs( const v : TVectorGL) : TVectorGL;
14228: Function glVectorAbs1( const v : TAffineVector) : TAffineVector;
14229: Procedure glSetMatrix( var dest : THomogeneousDblMatrix; const src : TMatrixGL);
14230: Procedure glSetMatrix1( var dest : TAffineMatrix; const src : TMatrixGL);
14231: Procedure glSetMatrix2( var dest : TMatrixGL; const src : TAffineMatrix);
14232: Procedure glSetMatrixRow( var dest : TMatrixGL; rowNb : Integer; const aRow : TVectorGL);
14233: Function glCreateScaleMatrix( const v : TAffineVector) : TMatrixGL;
14234: Function glCreateScaleMatrix1( const v : TVectorGL) : TMatrixGL;
14235: Function glCreateTranslationMatrix( const V : TAffineVector) : TMatrixGL;
14236: Function glCreateTranslationMatrix1( const v : TVectorGL) : TMatrixGL;
14237: Function glCreateScaleAndTranslationMatrix( const scale, offset : TVectorGL) : TMatrixGL;
14238: Function glCreateRotationMatrixX( const sine, cosine : Single) : TMatrixGL;
14239: Function glCreateRotationMatrixX1( const angle : Single) : TMatrixGL;
14240: Function glCreateRotationMatrixY( const sine, cosine : Single) : TMatrixGL;
14241: Function glCreateRotationMatrixY1( const angle : Single) : TMatrixGL;
14242: Function glCreateRotationMatrixZ( const sine, cosine : Single) : TMatrixGL;
14243: Function glCreateRotationMatrixZ1( const angle : Single) : TMatrixGL;
14244: Function glCreateRotationMatrix( const anAxis : TAffineVector; angle : Single) : TMatrixGL;
14245: Function glCreateRotationMatrix1( const anAxis : TVectorGL; angle : Single) : TMatrixGL;
14246: Function glCreateAffineRotationMatrix( const anAxis : TAffineVector; angle:Single):TAffineMatrix
14247: Function glMatrixMultiply( const M1, M2 : TAffineMatrix) : TAffineMatrix;
14248: Function glMatrixMultiply1( const M1, M2 : TMatrixGL) : TMatrixGL;
14249: Procedure glMatrixMultiply2( const M1, M2 : TMatrixGL; var MResult : TMatrixGL);
14250: Function glVectorTransform( const V : TVectorGL; const M : TMatrixGL) : TVectorGL;
14251: Function glVectorTransform1( const V : TVectorGL; const M : TAffineMatrix) : TVectorGL;
14252: Function glVectorTransform2( const V : TAffineVector; const M : TMatrixGL) : TAffineVector;
14253: Function glVectorTransform3( const V : TAffineVector; const M : TAffineMatrix) : TAffineVector;
14254: Function glMatrixDeterminant( const M : TAffineMatrix) : Single;
14255: Function glMatrixDeterminant1( const M : TMatrixGL) : Single;
14256: Procedure glAdjointMatrix( var M : TMatrixGL);
14257: Procedure glAdjointMatrix1( var M : TAffineMatrix);
14258: Procedure glScaleMatrix( var M : TAffineMatrix; const factor : Single);
14259: Procedure glScaleMatrix1( var M : TMatrixGL; const factor : Single);
14260: Procedure glTranslateMatrix( var M : TMatrixGL; const v : TAffineVector);
14261: Procedure glTranslateMatrix1( var M : TMatrixGL; const v : TVectorGL);
14262: Procedure glNormalizeMatrix( var M : TMatrixGL);
14263: Procedure glTransposeMatrix( var M : TAffineMatrix);
14264: Procedure glTransposeMatrix1( var M : TMatrixGL);
14265: Procedure glInvertMatrix( var M : TMatrixGL);
14266: Procedure glInvertMatrix1( var M : TAffineMatrix);
14267: Function glAnglePreservingMatrixInvert( const mat : TMatrixGL) : TMatrixGL
14268: Function glMatrixDecompose( const M : TMatrixGL; var Tran : TTransformations) : Boolean
14269: Function glPlaneMake( const p1, p2, p3 : TAffineVector) : THmgPlane;
14270: Function glPlaneMake1( const p1, p2, p3 : TVectorGL) : THmgPlane;
14271: Function glPlaneMake2( const point, normal : TAffineVector) : THmgPlane;
14272: Function glPlaneMake3( const point, normal : TVectorGL) : THmgPlane;
14273: Procedure glSetPlane( var dest : TDoubleHmgPlane; const src : THmgPlane)
14274: Procedure glNormalizePlane( var plane : THmgPlane)
14275: Function glPlaneEvaluatePoint( const plane : THmgPlane; const point : TAffineVector) : Single;
14276: Function glPlaneEvaluatePoint1( const plane : THmgPlane; const point : TVectorGL) : Single;
14277: Function glCalcPlaneNormal( const p1, p2, p3 : TAffineVector) : TAffineVector;
14278: Procedure glCalcPlaneNormal1( const p1, p2, p3 : TAffineVector; var vr : TAffineVector);
14279: Procedure glCalcPlaneNormal2( const p1, p2, p3 : TVectorGL; var vr : TAffineVector);
14280: Function glPointsInHalfSpace( const point, planePoint, planeNormal : TVectorGL) : Boolean;
14281: Function glPointIsInHalfSpace1( const point, planePoint, planeNormal : TAffineVector) : Boolean;
14282: Function glPointPlaneDistance( const point, planePoint, planeNormal : TVectorGL) : Single;
14283: Function glPointPlaneDistance1( const point, planePoint, planeNormal : TAffineVector) : Single;
14284: Function glPointSegmentClosestPoint( const point, segmentStart, segmentStop:TAffineVector):TAffineVector
14285: Function glPointSegmentDistance( const point, segmentStart, segmentStop : TAffineVector) : single
14286: Function glPointLineClosestPoint( const point, linePoint, lineDirection : TAffineVector) : TAffineVector
14287: Function glPointLineDistance( const point, linePoint, lineDirection : TAffineVector) : Single
14288: Procedure SglegmentSegmentClosestPoint( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector; var
Segment0Closest, Segment1Closest : TAffineVector)
14289: Function glSegmentDistance( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector) : single
14290: TEulerOrder', '( eulXYZ, eulXZY, eulYXZ, eulYZX, eulZXY, eulZYX)
14291: Function glQuaternionMake( const Imag : array of Single; Real : Single) : TQuaternion
14292: Function glQuaternionConjugate( const Q : TQuaternion) : TQuaternion
14293: Function glQuaternionMagnitude( const Q : TQuaternion) : Single
14294: Procedure glNormalizeQuaternion( var Q : TQuaternion)
14295: Function glQuaternionFromPoints( const V1, V2 : TAffineVector) : TQuaternion
14296: Procedure glQuaternionToPoints( const Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
14297: Function glQuaternionFromMatrix( const mat : TMatrixGL) : TQuaternion
14298: Function glQuaternionToMatrix( quat : TQuaternion) : TMatrixGL
14299: Function glQuaternionToAffineMatrix( quat : TQuaternion) : TAffineMatrix
14300: Function glQuaternionFromAngleAxis( const angle : Single; const axis : TAffineVector) : TQuaternion
14301: Function glQuaternionFromRollPitchYaw( const r, p, y : Single) : TQuaternion
14302: Function glQuaternionFromEuler( const x, y, z : Single; eulerOrder : TEulerOrder) : TQuaternion
14303: Function glQuaternionMultiply( const qL, qR : TQuaternion) : TQuaternion
14304: Function glQuaternionSlerp( const QStart, QEnd : TQuaternion; Spin : Integer; t : Single) : TQuaternion;
14305: Function glQuaternionSlerp1( const source, dest : TQuaternion; const t : Single) : TQuaternion;
14306: Function glLnNP1( X : Extended) : Extended

```

```

14307: Function glLog10( X : Extended) : Extended
14308: Function glLog2( X : Extended) : Extended;
14309: Function glLog2l( X : Single) : Single;
14310: Function glLogN( Base, X : Extended) : Extended
14311: Function glIntPower( Base : Extended; Exponent : Integer) : Extended
14312: Function glPower( const Base, Exponent : Single) : Single;
14313: Function glPowerl( Base : Single; Exponent : Integer) : Single;
14314: Function glDegToRad( const Degrees : Extended) : Extended;
14315: Function glDegToRadl( const Degrees : Single) : Single;
14316: Function glRadToDeg( const Radians : Extended) : Extended;
14317: Function glRadToDegl( const Radians : Single) : Single;
14318: Function glNormalizeAngle( angle : Single) : Single
14319: Function glNormalizeDegAngle( angle : Single) : Single
14320: Procedure glSinCos( const Theta : Extended; var Sin, Cos : Extended);
14321: Procedure glSinCosl( const Theta : Double; var Sin, Cos : Double);
14322: Procedure glSinCos( const Theta : Single; var Sin, Cos : Single);
14323: Procedure glSinCosl( const theta, radius : Double; var Sin, Cos : Extended);
14324: Procedure glSinCos2( const theta, radius : Double; var Sin, Cos : Double);
14325: Procedure glSinCos3( const theta, radius : Single; var Sin, Cos : Single);
14326: Procedure glPrepareSinCosCache( var s, c : array of Single; startAngle, stopAngle : Single)
14327: Function glArcCos( const X : Extended) : Extended;
14328: Function glArcCosl( const x : Single) : Single;
14329: Function glArcSin( const X : Extended) : Extended;
14330: Function glArcSinl( const X : Single) : Single;
14331: Function glArcTan2l( const Y, X : Extended) : Extended;
14332: Function glArcTan2l( const Y, X : Single) : Single;
14333: Function glFastArcTan2( y, x : Single) : Single
14334: Function glTan( const X : Extended) : Extended;
14335: Function glTanl( const X : Single) : Single;
14336: Function glCoTan( const X : Extended) : Extended;
14337: Function glCoTanl( const X : Single) : Single;
14338: Function glSinh( const x : Single) : Single;
14339: Function glSinh1( const x : Double) : Double;
14340: Function glCosh( const x : Single) : Single;
14341: Function glCosh1( const x : Double) : Double;
14342: Function glRSqrt( v : Single) : Single
14343: Function glRLength( x, y : Single) : Single
14344: Function glISqrt( i : Integer) : Integer
14345: Function glLLength( x, y : Integer) : Integer;
14346: Function glLLengthl( x, y, z : Integer) : Integer;
14347: Procedure glRegisterBasedExp
14348: Procedure glRandomPointOnSphere( var p : TAffineVector)
14349: Function glRoundInt( v : Single) : Single;
14350: Function glRoundIntl( v : Extended) : Extended;
14351: Function glTrunc( v : Single) : Integer;
14352: Function glTrunc64( v : Extended) : Int64;
14353: Function glInt( v : Single) : Single;
14354: Function glIntl( v : Extended) : Extended;
14355: Function glFrac( v : Single) : Single;
14356: Function glFrac1( v : Extended) : Extended;
14357: Function glRound( v : Single) : Integer;
14358: Function glRound64( v : Single) : Int64;
14359: Function glRound64l( v : Extended) : Int64;
14360: Function glTrunc( X : Extended) : Int64
14361: Function glRound( X : Extended) : Int64
14362: Function glFrac( X : Extended) : Extended
14363: Function glCeil( v : Single) : Integer;
14364: Function glCeil64( v : Extended) : Int64;
14365: Function glFloor( v : Single) : Integer;
14366: Function glFloor64( v : Extended) : Int64;
14367: Function glScaleAndRound( i : Integer; var s : Single) : Integer
14368: Function glSign( x : Single) : Integer
14369: Function glIsInRange( const x, a, b : Single) : Boolean;
14370: Function glIsInRangel( const x, a, b : Double) : Boolean;
14371: Function glIsInCube( const p, d : TAffineVector) : Boolean;
14372: Function glIsInCubel( const p, d : TVectorGL) : Boolean;
14373: //Function MinFloat( values : PSingleArray; nbItems : Integer) : Single;
14374: //Function MinFloat1( values : PDoubleArray; nbItems : Integer) : Double;
14375: //Function MinFloat2( values : PExtendedArray; nbItems : Integer) : Extended;
14376: Function glMinFloat3( const v1, v2 : Single) : Single;
14377: Function glMinFloat4( const v : array of Single) : Single;
14378: Function glMinFloat5( const v1, v2 : Double) : Double;
14379: Function glMinFloat6( const v1, v2 : Extended) : Extended;
14380: Function glMinFloat7( const v1, v2, v3 : Single) : Single;
14381: Function glMinFloat8( const v1, v2, v3 : Double) : Double;
14382: Function glMinFloat9( const v1, v2, v3 : Extended) : Extended;
14383: //Function MaxFloat10( values : PSingleArray; nbItems : Integer) : Single;
14384: //Function MaxFloat( values : PDoubleArray; nbItems : Integer) : Double;
14385: //Function MaxFloat1( values : PExtendedArray; nbItems : Integer) : Extended;
14386: Function glMaxFloat2( const v : array of Single) : Single;
14387: Function glMaxFloat3( const v1, v2 : Single) : Single;
14388: Function glMaxFloat4( const v1, v2 : Double) : Double;
14389: Function glMaxFloat5( const v1, v2 : Extended) : Extended;
14390: Function glMaxFloat6( const v1, v2, v3 : Single) : Single;
14391: Function glMaxFloat7( const v1, v2, v3 : Double) : Double;
14392: Function glMaxFloat8( const v1, v2, v3 : Extended) : Extended;
14393: Function glMinInteger9( const v1, v2 : Integer) : Integer;
14394: Function glMinInteger( const v1, v2 : Cardinal) : Cardinal;
14395: Function glMaxInteger( const v1, v2 : Integer) : Integer;

```

```

14396: Function glMaxInteger1( const v1, v2 : Cardinal) : Cardinal;
14397: Function glTriangleArea( const p1, p2, p3 : TAffineVector) : Single;
14398: //Function PolygonArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14399: Function glTriangleSignedArea( const p1, p2, p3 : TAffineVector) : Single;
14400: //Function PolygonSignedArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14401: //Procedure ScaleFloatArray( values : PSingleArray; nb : Integer; var factor : Single);
14402: Procedure glScaleFloatArray( var values : TSingleArray; factor : Single);
14403: //Procedure OffsetFloatArray( values : PSingleArray; nb : Integer; var delta : Single);
14404: Procedure glOffsetFloatArray1( var values : array of Single; delta : Single);
14405: //Procedure OffsetFloatArray2( valuesDest, valuesDelta : PSingleArray; nb : Integer);
14406: Function glMaxXYZComponent( const v : TVectorGL) : Single;
14407: Function glMaxXYZComponent1( const v : TAffineVector) : single;
14408: Function glMinXYZComponent( const v : TVectorGL) : Single;
14409: Function glMinXYZComponent1( const v : TAffineVector) : single;
14410: Function glMaxAbsXYZComponent( v : TVectorGL) : Single
14411: Function glMinAbsXYZComponent( v : TVectorGL) : Single
14412: Procedure glMaxVector( var v : TVectorGL; const v1 : TVectorGL);
14413: Procedure glMaxVector1( var v : TAffineVector; const v1 : TAffineVector);
14414: Procedure glMinVector( var v : TVectorGL; const v1 : TVectorGL);
14415: Procedure glMinVector1( var v : TAffineVector; const v1 : TAffineVector);
14416: Procedure glSortArrayAscending( var a : array of Extended);
14417: Function glClampValue( const aValue, aMin, aMax : Single) : Single;
14418: Function glClampValue1( const aValue, aMin : Single) : Single;
14419: Function glGeometryOptimizationMode : String
14420: Procedure glBeginFPUOnlySection
14421: Procedure glEndFPUOnlySection
14422: Function glConvertRotation( const Angles : TAffineVector) : TVectorGL
14423: Function glMakeAffineDblVector( var v : array of Double) : TAffineDblVector
14424: Function glMakeDblVector( var v : array of Double) : THomogeneousDblVector
14425: Function glVectorAffineDblToFlt( const v : TAffineDblVector) : TAffineVector
14426: Function glVectorDblToFlt( const v : THomogeneousDblVector) : THomogeneousVector
14427: Function glVectorAffineFltToDbl( const v : TAffineVector) : TAffineDblVector
14428: Function glVectorFltToDbl( const v : TVectorGL) : THomogeneousDblVector
14429: Function glPointInPolygon( var xp, yp : array of Single; x, y : Single) : Boolean
14430: Procedure glDivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
14431: Function glTurn( const Matrix : TMatrixGL; angle : Single) : TMatrixGL;
14432: Function glTurn1( const Matrix : TMatrixGL; const MasterUp:TAffineVector;Angle:Single):TMatrixGL;
14433: Function glPitch( const Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
14434: Function glPitch1( const Matrix:TMatrixGL;const MasterRight:TAffineVector;Angle:Single):TMatrixGL;
14435: Function glRoll( const Matrix: TMatrixGL; Angle : Single) : TMatrixGL;
14436: Function glRoll1(const Matrix: TMatrixGL;const MasterDirection:TAffineVector;Angle: Single): TMatrixGL;
14437: Function glRayCastMinDistToPoint( const rayStart, rayVector: TVectorGL;const point:TVectorGL):Single
14438: Function glRayCastIntersectsSphere( const rayStart, rayVector : TVectorGL; const
sphereCenter:TVectorGL;const sphereRadius : Single) : Boolean;
14439: Function glRayCastSphereIntersect( const rayStart, rayVector : TVectorGL; const sphereCenter:TVectorGL;
const sphereRadius : Single; var i1, i2 : TVectorGL) : Integer;
14440: Function glSphereVisibleRadius( distance, radius : Single) : Single
14441: Function glExtractFrustumFromModelViewProjection( const modelViewProj : TMatrixGL) : TFrustum
14442: Function glIsVolumeClipped( const objPos : TVectorGL; const objRadius : Single; const rcci :
TRenderContextClippingInfo) : Boolean;
14443: Function glIsVolumeClipped1( const objPos : TAffineVector; const objRadius : Single; const rcci :
TRenderContextClippingInfo) : Boolean;
14444: Function glIsVolumeClipped2(const min,max:TAffineVector;const rcci : TRenderContextClippingInfo) : Bool;
14445: Function glIsVolumeClipped3(const objPos:TAffineVector;const objRadius:Single;const
Frustum:TFrustum):Bool;
14446: Function glMakeParallelProjectionMatrix( const plane : THmgPlane; const dir : TVectorGL) : TMatrixGL
14447: Function glMakeShadowMatrix( const planePoint, planeNormal, lightPos : TVectorGL) : TMatrixGL
14448: Function glMakeReflectionMatrix( const planePoint, planeNormal : TAffineVector) : TMatrixGL
14449: Function glPackRotationMatrix( const mat : TMatrixGL) : TPackedRotationMatrix
14450: Function glUnPackRotationMatrix( const packedMatrix : TPackedRotationMatrix) : TMatrixGL
14451: 'cPI','Single').setExtended( 3.141592654);
14452: 'cPIdiv180','Single').setExtended( 0.017453292);
14453: 'c180divPI','Single').setExtended( 57.29577951);
14454: 'c2PI','Single').setExtended( 6.283185307);
14455: 'cPIdiv2','Single').setExtended(1.570796326);
14456: 'cPIdiv4','Single').setExtended( 0.785398163);
14457: 'c3PIdiv4','Single').setExtended( 2.35619449);
14458: 'cInv2PI','Single').setExtended( 1 / 6.283185307);
14459: 'cInv360','Single').setExtended( 1 / 360);
14460: 'c180','Single').setExtended( 180);
14461: 'c360','Single').setExtended( 360);
14462: 'cOneHalf','Single').setExtended( 0.5);
14463: 'cLn10','Single').setExtended( 2.302585093);
14464: {'MinSingle','Extended').setExtended( 1.5e-45);
14465: 'MaxSingle','Extended').setExtended( 3.4e+38);
14466: 'MinDouble','Extended').setExtended( 5.0e-324);
14467: 'MaxDouble','Extended').setExtended( 1.7e+308);
14468: 'MinExtended','Extended').setExtended( 3.4e-4932);
14469: 'MaxExtended','Extended').setExtended( 1.1e+4932);
14470: 'MinComp','Extended').setExtended( - 9.223372036854775807e+18);
14471: 'MaxComp','Extended').setExtended( 9.223372036854775807e+18);}
14472: end;
14473:
14474: procedure SIRegister_GLVectorFileObjects(CL: TPSPascalCompiler);
14475: begin
14476:   AddClassN(FindClass('TOBJECT'),'TMeshObjectList
14477:   (FindClass('TOBJECT'),'TFaceGroups
14478:   TMeshAutoCentering', '( macCenterX, macCenterY, macCenterZ, macUseBarycenter )
14479:   TMeshAutoCenterings', 'set of TMeshAutoCentering

```



```

14480: TMeshObjectMode', '( momTriangles, momTriangleStrip, momFaceGroups )
14481: SIRegister_TBaseMeshObject(CL);
14482: (FindClass('TOBJECT'),'TSkeletonFrameList
14483: TSkeletonFrameTransform', '( sftRotation, sftQuaternion )
14484: SIRegister_TSkeletonFrame(CL);
14485: SIRegister_TSkeletonFrameList(CL);
14486: (FindClass('TOBJECT'),'TSkeleton
14487: (FindClass('TOBJECT'),'TSkeletonBone
14488: SIRegister_TSkeletonBoneList(CL);
14489: SIRegister_TSkeletonRootBoneList(CL);
14490: SIRegister_TSkeletonBone(CL);
14491: (FindClass('TOBJECT'),'TSkeletonColliderList
14492: SIRegister_TSkeletonCollider(CL);
14493: SIRegister_TSkeletonColliderList(CL);
14494: (FindClass('TOBJECT'),'TGLBaseMesh
14495: TBlendedLerpInfo', 'record frameIndex1 : Integer; frameIndex2 : '
14496:   + 'Integer; lerpFactor : Single; weight : Single; externalPositions : TAffine'
14497:   + 'VectorList; externalRotations : TAffineVectorList; externalQuaternions : T'
14498:   + 'QuaternionList; end
14499: SIRegister_TSkeleton(CL);
14500: TMeshObjectRenderingOption', '( moroGroupByMaterial )
14501: TMeshObjectRenderingOptions', 'set of TMeshObjectRenderingOption
14502: SIRegister_TMeshObject(CL);
14503: SIRegister_TMeshObjectList(CL);
14504: //TMeshObjectListClass', 'class of TMeshObjectList
14505: (FindClass('TOBJECT'),'TMeshMorphTargetList
14506: SIRegister_TMeshMorphTarget(CL);
14507: SIRegister_TMeshMorphTargetList(CL);
14508: SIRegister_TMorphableMeshObject(CL);
14509: TVertexBoneWeight', 'record BoneID : Integer; Weight : Single; end
14510: //TVertexBoneWeightArray', '^TVertexBoneWeightArray // will not work
14511: //TVerticesBoneWeights', '^TVerticesBoneWeights // will not work
14512: TVertexBoneWeightDynArray', 'array of TVertexBoneWeight
14513: SIRegister_TSkeletonMeshObject(CL);
14514: SIRegister_TFaceGroup(CL);
14515: TFaceGroupMeshMode', '( fgmTriangles, fgmTriangleStrip, fgmFl'
14516:   + 'atTriangles, fgmTriangleFan, fgmQuads )
14517: SIRegister_TFGVertexIndexList(CL);
14518: SIRegister_TFGVertexNormalTexIndexList(CL);
14519: SIRegister_TFGIndexTexCoordList(CL);
14520: SIRegister_TFaceGroups(CL);
14521: TMeshNormalsOrientation', '( mnoDefault, mnoInvert )
14522: SIRegister_TVectorFile(CL);
14523: //TVectorFileClass', 'class of TVectorFile
14524: SIRegister_TGLGLSMVectorFile(CL);
14525: SIRegister_TGLBaseMesh(CL);
14526: SIRegister_TGLFreeForm(CL);
14527: TGLActorOption', '( aoSkeletonNormalizeNormals )
14528: TGLActorOptions', 'set of TGLActorOption
14529: 'cDefaultGLActorOptions', 'LongInt'.Value.ts32:= ord(aoSkeletonNormalizeNormals);
14530: (FindClass('TOBJECT'),'TGLActor
14531: TActorAnimationReference', '( aarMorph, aarSkeleton, aarNone )
14532: SIRegister_TActorAnimation(CL);
14533: TActorAnimationName', 'String
14534: SIRegister_TActorAnimations(CL);
14535: SIRegister_TGLBaseAnimationController(CL);
14536: SIRegister_TGLAnimationController(CL);
14537: TActorFrameInterpolation', '( afpNone, afpLinear )
14538: TActorAnimationMode', '( aamNone, aamPlayOnce, aamLoop, aamBounc'
14539:   + 'eForward, aamBounceBackward, aamLoopBackward, aamExternal )
14540: SIRegister_TGLActor(CL);
14541: SIRegister_TVectorFileFormat(CL);
14542: SIRegister_TVectorFileFormatsList(CL);
14543: (FindClass('TOBJECT'),'EInvalidVectorFile
14544: Function GetVectorFileFormats : TVectorFileFormatsList
14545: Function VectorFileFormatsFilter : String
14546: Function VectorFileFormatsSaveFilter : String
14547: Function VectorFileFormatExtensionByIndex( index : Integer) : String
14548: Procedure RegisterVectorFileFormat( const aExtension, aDescription : String; aClass : TVectorFileClass)
14549: Procedure UnregisterVectorFileClass( aClass : TVectorFileClass)
14550: end;
14551:
14552: procedure SIRegister_AxCtrls(CL: TPSPascalCompiler);
14553: begin
14554:   'Class_DColorPropPage', 'TGUID').SetString( '{5CFF5D59-5946-11D0-BDEF-00A024D1875C}'
14555:   'Class_DFontPropPage', 'TGUID').SetString( '{5CFF5D5B-5946-11D0-BDEF-00A024D1875C}'
14556:   'Class_DPicturePropPage', 'TGUID').SetString( '{5CFF5D5A-5946-11D0-BDEF-00A024D1875C}'
14557:   'Class_DStringPropPage', 'TGUID').SetString( '{F42D677E-754B-11D0-BDFB-00A024D1875C}'
14558:   SIRegister_TOLEStream(CL);
14559:   (FindClass('TOBJECT'),'TConnectionPoints
14560:   TConnectionKind', '( ckSingle, ckMulti )
14561:   SIRegister_TConnectionPoint(CL);
14562:   SIRegister_TConnectionPoints(CL);
14563:   TDefinePropertyPage', 'Procedure ( const GUID : TGUID)
14564:   (FindClass('TOBJECT'),'TActiveXControlFactory
14565:   SIRegister_TActiveXControl(CL);
14566:   //TActiveXControlClass', 'class of TActiveXControl
14567:   SIRegister_TActiveXControlFactory(CL);
14568:   SIRegister_TActiveFormControl(CL);

```

```

14569: SIRegister_TActiveForm(CL);
14570: //TActiveFormClass', 'class of TActiveForm
14571: SIRegister_TActiveFormFactory(CL);
14572: (FindClass('TObject'), 'TPropertyPageImpl
14573: SIRegister_TPropertyPage(CL);
14574: //TPropertyPageClass', 'class of TPropertyPage
14575: SIRegister_TPropertyPageImpl(CL);
14576: SIRegister_TActiveXPropertyPage(CL);
14577: SIRegister_TActiveXPropertyPageFactory(CL);
14578: SIRegister_TCustomAdapter(CL);
14579: SIRegister_TAdapterNotifier(CL);
14580: SIRegister_IFontAccess(CL);
14581: SIRegister_TFontAdapter(CL);
14582: SIRegister_IPictureAccess(CL);
14583: SIRegister_TPictureAdapter(CL);
14584: SIRegister_TOLEGraphic(CL);
14585: SIRegister_TStringsAdapter(CL);
14586: SIRegister_TReflectorWindow(CL);
14587: Procedure EnumDispatchProperties(Dispatch: IDispatch; PropType: TGUID; VTCode: Int; PropList: TStrings);
14588: Procedure GetOleFont( Font : TFont; var OleFont : IFontDisp)
14589: Procedure SetOleFont( Font : TFont; OleFont : IFontDisp)
14590: Procedure GetOlePicture( Picture : TPicture; var OlePicture : IPictureDisp)
14591: Procedure SetOlePicture( Picture : TPicture; OlePicture : IPictureDisp)
14592: Procedure GetOleStrings( Strings : TStrings; var OleStrings : IStrings)
14593: Procedure SetOleStrings( Strings : TStrings; OleStrings : IStrings)
14594: Function ParkingWindow : HWND
14595: end;
14596:
14597: procedure SIRegister_synaip(CL: TPSPascalCompiler);
14598: begin
14599: // Tip6Bytes = array [0..15] of Byte;
14600: {binary form of IPv6 address (for string conversion routines)}
14601: // Tip6Words = array [0..7] of Word;
14602: AddTypeS('Tip6Bytes', 'array [0..15] of Byte;');
14603: AddTypeS('Tip6Words', 'array [0..7] of Word;');
14604: AddTypeS('TIPAdr', 'record Oct1 : Byte; Oct2 : Byte; Oct3 : Byte; Oct4 : Byte; end;');
14605: Function synaIsIP( const Value : string) : Boolean;
14606: Function synaIsIP6( const Value : string) : Boolean;
14607: Function synaIPToID( Host : string) : AnsiString;
14608: Function synaStrToIp6( value : string) : Tip6Bytes;
14609: Function synaIp6ToStr( value : Tip6Bytes) : string;
14610: Function synaStrToIp( value : string) : integer;
14611: Function synaIpToStr( value : integer) : string;
14612: Function synaReverseIP( Value : AnsiString) : AnsiString;
14613: Function synaReverseIP6( Value : AnsiString) : AnsiString;
14614: Function synaExpandIP6( Value : AnsiString) : AnsiString;
14615: Function xStrToIP( const Value : String) : TIPAdr;
14616: Function xIPToStr( const Adresse : TIPAdr) : String;
14617: Function IPToCardinal( const Adresse : TIPAdr) : Cardinal;
14618: Function CardinalToIP( const Value : Cardinal) : TIPAdr;
14619: end;
14620:
14621: procedure SIRegister_synacode(CL: TPSPascalCompiler);
14622: begin
14623: AddTypeS('TSpecials', 'set of Char');
14624: Const('SpecialChar', 'TSpecials').SetSet( '=<>:;,@/?\"_');
14625: Const('URLFullSpecialChar', 'TSpecials').SetSet( ';/?:@=&#+');
14626: Const('TableBase64', 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=');
14627: Const('TableBase64mod', 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+,=');
14628: Const('TableUU'('!'#$%&'()*+,-./0123456789;=<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_');
14629: Const('TableXX'('+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz');
14630: Function DecodeTriplet( const Value : AnsiString; Delimiter : Char; Specials : TSpecials) : AnsiString;
14631: Function DecodeQuotedPrintable( const Value : AnsiString) : AnsiString;
14632: Function DecodeURL( const Value : AnsiString) : AnsiString;
14633: Function EncodeTriplet( const Value : AnsiString; Delimiter : Char; Specials : TSpecials) : AnsiString;
14634: Function EncodeQuotedPrintable( const Value : AnsiString) : AnsiString;
14635: Function EncodeSafeQuotedPrintable( const Value : AnsiString) : AnsiString;
14636: Function EncodeURLElement( const Value : AnsiString) : AnsiString;
14637: Function DecodeURL( const Value : AnsiString) : AnsiString;
14638: Function Decode4to3( const Value, Table : AnsiString) : AnsiString;
14639: Function Decode4to3Ex( const Value, Table : AnsiString) : AnsiString;
14640: Function Encode3to4( const Value, Table : AnsiString) : AnsiString;
14641: Function synDecodeBase64( const Value : AnsiString) : AnsiString;
14642: Function synEncodeBase64( const Value : AnsiString) : AnsiString;
14643: Function DecodeBase64mod( const Value : AnsiString) : AnsiString;
14644: Function EncodeBase64mod( const Value : AnsiString) : AnsiString;
14645: Function DecodeUU( const Value : AnsiString) : AnsiString;
14646: Function EncodeUU( const Value : AnsiString) : AnsiString;
14647: Function DecodeXX( const Value : AnsiString) : AnsiString;
14648: Function DecodeYEnc( const Value : AnsiString) : AnsiString;
14649: Function UpdateCrc32( Value : Byte; Crc32 : Integer) : Integer;
14650: Function synCrc32( const Value : AnsiString) : Integer;
14651: Function UpdateCrc16( Value : Byte; Crc16 : Word) : Word;
14652: Function Crc16( const Value : AnsiString) : Word;
14653: Function synMD5( const Value : AnsiString) : AnsiString;
14654: Function HMAC_MD5( Text, Key : AnsiString) : AnsiString;
14655: Function MD5LongHash( const Value : AnsiString; Len : integer) : AnsiString;
14656: Function synSHA1( const Value : AnsiString) : AnsiString;
14657: Function HMAC_SHA1( Text, Key : AnsiString) : AnsiString;

```

```

14658:   Function SHA1LongHash( const Value : AnsiString; Len : integer) : AnsiString');
14659:   Function synMD4( const Value : AnsiString) : AnsiString');
14660: end;
14661:
14662: procedure SIRegister_synachar(CL: TPSPascalCompiler);
14663: begin
14664:   AddTypeS('TTimeChar', '( ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, I'
14665:   + 'SO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, IS'
14666:   + 'O_8859_13, ISO_8859_14, ISO_8859_15, CP1250, CP1251, CP1252, CP1253, CP125'
14667:   + '4, CP1255, CP1256, CP1257, CP1258, KOI8_R, CP895, CP852, UCS_2, UCS_4, UTF'
14668:   + '_8, UTF_7, UTF_7mod, UCS_2LE, UCS_4LE, UTF_16, UTF_16LE, UTF_32, UTF_32LE,'
14669:   + 'C99, JAVA, ISO_8859_16, KOI8_U, KOI8_RU, CP862, CP866, MAC, MACCE, MACICE'
14670:   + ', MACCRO, MACRO, MACCYR, MACUK, MACGR, MACTU, MACHEB, MACAR, MACTH, ROMAN8'
14671:   + ', NEXTSTEP, ARMSCII, GEORGIAN_AC, GEORGIAN_PS, KOI8_T, MULELAO, CP1133, T'
14672:   + 'IS620, CP874, VISCII, TCVN, ISO_IR_14, JIS_X0201, JIS_X0208, JIS_X0212, GB'
14673:   + '1988_80, GB2312_80, ISO_IR_165, ISO_IR_149, EUC_JP, SHIFT_JIS, CP932, ISO_'
14674:   + '2022_JP, ISO_2022_JP1, ISO_2022_JP2, GB2312, CP936, GB18030, ISO_2022_CN, '
14675:   + 'ISO_2022_CNE, HZ, EUC_TW, BIG5, CP950, BIG5_HKSCS, EUC_KR, CP949, CP1361, '
14676:   + 'ISO_2022_KR, CP737, CP775, CP853, CP855, CP857, CP858, CP860, CP861, CP863'
14677:   + ', CP864, CP865, CP869, CP1125 )');
14678:   AddTypeS('TTimeSetChar', 'set of TTimeChar');
14679:   Function CharSetConversion(const Value:AnsiString;CharFrom:TTimeChar;CharTo:TTimeChar):AnsiString;
14680:   Function CharSetConversionEx(const Value:AnsiString;CharFrom:TTimeChar;CharTo:TTimeChar; const
   TransformTable : array of Word) : AnsiString');
14681:   Function CharSetConversionTrans( Value : AnsiString; CharFrom : TTimeChar; CharTo : TTimeChar; const
   TransformTable : array of Word; Translit : Boolean) : AnsiString');
14682:   Function GetCurCP : TTimeChar');
14683:   Function GetCurOEMCP : TTimeChar');
14684:   Function GetCPFromID( Value : AnsiString) : TTimeChar');
14685:   Function GetIDFromCP( Value : TTimeChar) : AnsiString');
14686:   Function NeedCharSetConversion( const Value : AnsiString) : Boolean');
14687:   Function IdealCharSetCoding(const Value:AnsiString;CharFrom:TTimeChar;CharTo:TTimeSetChar):TTimeChar;
14688:   Function GetBOM( Value : TTimeChar) : AnsiString');
14689:   Function StringToWide( const Value : AnsiString) : WideString');
14690:   Function WideToString( const Value : WideString) : AnsiString');
14691: end;
14692:
14693: procedure SIRegister_synamisc(CL: TPSPascalCompiler);
14694: begin
14695:   AddTypeS('TProxySetting', 'record Host : string; Port : string; Bypass : string; end');
14696:   Procedure WakeOnLan( MAC, IP : string');
14697:   Function GetDNS : string');
14698:   Function GetIEProxy( protocol : string) : TProxySetting');
14699:   Function GetLocalIPs : string');
14700: end;
14701:
14702:
14703: procedure SIRegister_synaser(CL: TPSPascalCompiler);
14704: begin
14705:   AddConstantN('synCR','Char').SetString( #$0d);
14706:   Const('synLF','Char').SetString( #$0a);
14707:   Const('cSerialChunk','LongInt').SetInt( 8192);
14708:   Const('LockfileDirectory','String').SetString( '/var/lock');
14709:   Const('PortIsClosed','LongInt').SetInt( - 1);
14710:   Const('ErrAlreadyOwned','LongInt').SetInt( 9991);
14711:   Const('ErrAlreadyInUse','LongInt').SetInt( 9992);
14712:   Const('ErrWrongParameter','LongInt').SetInt( 9993);
14713:   Const('ErrPortNotOpen','LongInt').SetInt( 9994);
14714:   Const('ErrNoDeviceAnswer','LongInt').SetInt( 9995);
14715:   Const('ErrMaxBuffer','LongInt').SetInt( 9996);
14716:   Const('ErrTimeout','LongInt').SetInt( 9997);
14717:   Const('ErrNotRead','LongInt').SetInt( 9998);
14718:   Const('ErrFrame','LongInt').SetInt( 9999);
14719:   Const('ErrOverrun','LongInt').SetInt( 10000);
14720:   Const('ErrRxOver','LongInt').SetInt( 10001);
14721:   Const('ErrRxParity','LongInt').SetInt( 10002);
14722:   Const('ErrTxFull','LongInt').SetInt( 10003);
14723:   Const('dcb_Binary','LongWord').SetUInt( $00000001);
14724:   Const('dcb_ParityCheck','LongWord').SetUInt( $00000002);
14725:   Const('dcb_OutxCtsFlow','LongWord').SetUInt( $00000004);
14726:   Const('dcb_OutxDsrFlow','LongWord').SetUInt( $00000008);
14727:   Const('dcb_DtrControlMask','LongWord').SetUInt( $00000030);
14728:   Const('dcb_DtrControlDisable','LongWord').SetUInt( $00000000);
14729:   Const('dcb_DtrControlEnable','LongWord').SetUInt( $00000010);
14730:   Const('dcb_DtrControlHandshake','LongWord').SetUInt( $00000020);
14731:   Const('dcb_DsrSensitivity','LongWord').SetUInt( $00000040);
14732:   Const('dcb_TXContinueOnXoff','LongWord').SetUInt( $00000080);
14733:   Const('dcb_OutX','LongWord').SetUInt( $00000100);
14734:   Const('dcb_InX','LongWord').SetUInt( $00000200);
14735:   Const('dcb_ErrorChar','LongWord').SetUInt( $00000400);
14736:   Const('dcb_NullStrip','LongWord').SetUInt( $00000800);
14737:   Const('dcb_RtsControlMask','LongWord').SetUInt( $00003000);
14738:   Const('dcb_RtsControlDisable','LongWord').SetUInt( $00000000);
14739:   Const('dcb_RtsControlEnable','LongWord').SetUInt( $00001000);
14740:   Const('dcb_RtsControlHandshake','LongWord').SetUInt( $00002000);
14741:   Const('dcb_RtsControlToggle','LongWord').SetUInt( $00003000);
14742:   Const('dcb_AbortOnError','LongWord').SetUInt( $00004000);
14743:   Const('dcb_Reserveds','LongWord').SetUInt( $FFFF8000);
14744:   Const('synSB1','LongInt').SetInt( 0);

```

```

14745: Const('SBlandHalf','LongInt').SetInt( 1);
14746: Const('synSB2','LongInt').SetInt( 2);
14747: Const('synINVALID_HANDLE_VALUE','LongInt').SetInt( THandle ( - 1 ));
14748: Const('CS7fix','LongWord').SetUInt( $0000020);
14749: AddTypeS('synTDCB', 'record DCBlength : DWORD; BaudRate : DWORD; Flags : Long'
14750:   + 'int; wReserved : Word; XonLim : Word; XoffLim : Word; ByteSize : Byte; Par'
14751:   + 'ity : Byte; StopBits : Byte; XonChar : CHAR; XoffChar : CHAR; ErrorChar : '
14752:   + 'CHAR; EofChar : CHAR; EvtChar : CHAR; wReserved1 : Word; end');
14753: //AddTypeS('PDCB', '^TDCB // will not work');
14754: //Const('MaxRates','LongInt').SetInt( 18);
14755: //Const('MaxRates','LongInt').SetInt( 30);
14756: //Const('MaxRates','LongInt').SetInt( 19);
14757: Const('O_SYNC','LongWord').SetUInt( $0080);
14758: Const('synOK','LongInt').SetInt( 0);
14759: Const('synErr','LongInt').SetInt( integer ( - 1 ));
14760: AddTypeS('THookSerialReason', '( HR_SerialClose, HR_Connect, HR_CanRead, HR_CanWrite, HR_ReadCount,
HR_WriteCount, HR_Wait )');
14761: Type('THookSerialStatus',Procedure(Sender: TObject; Reason:THookSerialReason; const Value:string));
14762: SIRegister_ESynaSerError(CL);
14763: SIRegister_TBlockSerial(CL);
14764: Function GetSerialPortNames : string);
14765: end;
14766:
14767: procedure SIRegister_synaicnv(CL: TPSPascalCompiler);
14768: begin
14769: Const('DLLIconvName','String').SetString( 'libiconv.so');
14770: Const('DLLIconvName','String').SetString( 'iconv.dll');
14771: AddTypeS('size_t', 'Cardinal');
14772: AddTypeS('iconv_t', 'Integer');
14773: //AddTypeS('iconv_t', 'Pointer');
14774: AddTypeS('argptr', 'iconv_t');
14775: Function SynaIcnvOpen( const tocode, fromcode : Ansistring) : iconv_t);
14776: Function SynaIcnvOpenTranslit( const tocode, fromcode : Ansistring) : iconv_t);
14777: Function SynaIcnvOpenIgnore( const tocode, fromcode : Ansistring) : iconv_t);
14778: Function SynaIcnv( cd : iconv_t; inbuf : AnsiString; var outbuf : AnsiString) : integer);
14779: Function SynaIcnvClose( var cd : iconv_t) : integer);
14780: Function SynaIcnvCtl( cd : iconv_t; request : integer; argument : argptr) : integer);
14781: Function IsIcnvloaded : Boolean);
14782: Function InitIcnvInterface : Boolean);
14783: Function DestroyIcnvInterface : Boolean);
14784: Const('ICONV_TRIVIALP','LongInt').SetInt( 0);
14785: Const('ICONV_GET_TRANSLITERATE','LongInt').SetInt( 1);
14786: Const('ICONV_SET_TRANSLITERATE','LongInt').SetInt( 2);
14787: Const('ICONV_GET_DISCARD_ILSEQ','LongInt').SetInt( 3);
14788: Const('ICONV_SET_DISCARD_ILSEQ','LongInt').SetInt( 4);
14789: end;
14790:
14791: procedure SIRegister_pingsend(CL: TPSPascalCompiler);
14792: begin
14793: Const 'ICMP_ECHO','LongInt').SetInt( 8);
14794: Const('ICMP_ECHOREPLY','LongInt').SetInt( 0);
14795: Const('ICMP_UNREACH','LongInt').SetInt( 3);
14796: Const('ICMP_TIME_EXCEEDED','LongInt').SetInt( 11);
14797: Const('ICMP6_ECHO','LongInt').SetInt( 128);
14798: Const('ICMP6_ECHOREPLY','LongInt').SetInt( 129);
14799: Const('ICMP6_UNREACH','LongInt').SetInt( 1);
14800: Const('ICMP6_TIME_EXCEEDED','LongInt').SetInt( 3);
14801: AddTypeS('TICMPError', '( IE_NoError, IE_Other, IE_TTLExceed, IE_UnreachOt'
14802:   + 'her, IE_UnreachRoute, IE_UnreachAdmin, IE_UnreachAddr, IE_UnreachPort )');
14803: SIRegister_TPINGSend(CL);
14804: Function PingHost( const Host : string) : Integer);
14805: Function TraceRouteHost( const Host : string) : string);
14806: end;
14807:
14808: procedure SIRegister_asnluutil(CL: TPSPascalCompiler);
14809: begin
14810: AddConstantN('synASN1_BOOL','LongWord').SetUInt( $01);
14811: Const('synASN1_INT','LongWord').SetUInt( $02);
14812: Const('synASN1_OCTSTR','LongWord').SetUInt( $04);
14813: Const('synASN1_NULL','LongWord').SetUInt( $05);
14814: Const('synASN1_OBJID','LongWord').SetUInt( $06);
14815: Const('synASN1_ENUM','LongWord').SetUInt( $0a);
14816: Const('synASN1_SEQ','LongWord').SetUInt( $30);
14817: Const('synASN1_SETOF','LongWord').SetUInt( $31);
14818: Const('synASN1_IPADDR','LongWord').SetUInt( $40);
14819: Const('synASN1_COUNTER','LongWord').SetUInt( $41);
14820: Const('synASN1_GAUGE','LongWord').SetUInt( $42);
14821: Const('synASN1_TIMETICKS','LongWord').SetUInt( $43);
14822: Const('synASN1_OPAQUE','LongWord').SetUInt( $44);
14823: Function synASNEncOIDItem( Value : Integer) : AnsiString);
14824: Function synASNDecOIDItem( var Start : Integer; const Buffer : AnsiString) : Integer);
14825: Function synASNEncLen( Len : Integer) : AnsiString);
14826: Function synASNDecLen( var Start : Integer; const Buffer : AnsiString) : Integer);
14827: Function synASNEncInt( Value : Integer) : AnsiString);
14828: Function synASNEncUInt( Value : Integer) : AnsiString);
14829: Function synASNObject( const Data : AnsiString; ASNType : Integer) : AnsiString);
14830: Function synASNItem(var Start:Integer;const Buffer:AnsiString;var ValueType:Integer):AnsiString;
14831: Function synMibToId( Mib : String) : AnsiString);
14832: Function synIdToMib( const Id : AnsiString) : String);

```



```

14833: Function synIntMibToStr( const Value : AnsiString) : AnsiString);
14834: Function ASNdump( const Value : AnsiString) : AnsiString);
14835: Function GetMailServers( const DNSHost, Domain : AnsiString; const Servers:TStrings): Boolean);
14836: Function LDAPResultDump( const Value : TLdapResultList) : AnsiString);
14837: end;
14838:
14839: procedure SIRegister_ldapsend(CL: TPSPascalCompiler);
14840: begin
14841:   Const('cLDAPProtocol','String').SetString( '389');
14842:   Const('LDAP_ASN1_BIND_REQUEST','LongWord').SetUInt( $60);
14843:   Const('LDAP_ASN1_BIND_RESPONSE','LongWord').SetUInt( $61);
14844:   Const('LDAP_ASN1_UNBIND_REQUEST','LongWord').SetUInt( $42);
14845:   Const('LDAP_ASN1_SEARCH_REQUEST','LongWord').SetUInt( $63);
14846:   Const('LDAP_ASN1_SEARCH_ENTRY','LongWord').SetUInt( $64);
14847:   Const('LDAP_ASN1_SEARCH_DONE','LongWord').SetUInt( $65);
14848:   Const('LDAP_ASN1_SEARCH_REFERENCE','LongWord').SetUInt( $73);
14849:   Const('LDAP_ASN1_MODIFY_REQUEST','LongWord').SetUInt( $66);
14850:   Const('LDAP_ASN1_MODIFY_RESPONSE','LongWord').SetUInt( $67);
14851:   Const('LDAP_ASN1_ADD_REQUEST','LongWord').SetUInt( $68);
14852:   Const('LDAP_ASN1_ADD_RESPONSE','LongWord').SetUInt( $69);
14853:   Const('LDAP_ASN1_DEL_REQUEST','LongWord').SetUInt( $4A);
14854:   Const('LDAP_ASN1_DEL_RESPONSE','LongWord').SetUInt( $6B);
14855:   Const('LDAP_ASN1_MODIFYDN_REQUEST','LongWord').SetUInt( $6C);
14856:   Const('LDAP_ASN1_MODIFYDN_RESPONSE','LongWord').SetUInt( $6D);
14857:   Const('LDAP_ASN1_COMPARE_REQUEST','LongWord').SetUInt( $6E);
14858:   Const('LDAP_ASN1_COMPARE_RESPONSE','LongWord').SetUInt( $6F);
14859:   Const('LDAP_ASN1_ABANDON_REQUEST','LongWord').SetUInt( $70);
14860:   Const('LDAP_ASN1_EXT_REQUEST','LongWord').SetUInt( $77);
14861:   Const('LDAP_ASN1_EXT_RESPONSE','LongWord').SetUInt( $78);
14862:   SIRegister_TLDAPAttribute(CL);
14863:   SIRegister_TLDAPAttributeList(CL);
14864:   SIRegister_TLDAPResult(CL);
14865:   SIRegister_TLDAPResultList(CL);
14866:   AddTypeS('TLdapModifyOp', '( MO_Add, MO_Delete, MO_Replace )');
14867:   AddTypeS('TLdapSearchScope', '( SS_BaseObject, SS_SingleLevel, SS_WholeSubtree )');
14868:   AddTypeS('TLdapSearchAliases', '( SA_NeverDeref, SA_InSearching, SA_FindingBaseObj, SA_Always )');
14869:   SIRegister_TLDAPSend(CL);
14870: Function LDAPResultDump( const Value : TLdapResultList) : AnsiString);
14871: end;
14872:
14873:
14874: procedure SIRegister_slogsend(CL: TPSPascalCompiler);
14875: begin
14876:   Const('cSysLogProtocol','String').SetString( '514');
14877:   Const('FCL_Kernel','LongInt').SetInt( 0);
14878:   Const('FCL_UserLevel','LongInt').SetInt( 1);
14879:   Const('FCL_MailSystem','LongInt').SetInt( 2);
14880:   Const('FCL_System','LongInt').SetInt( 3);
14881:   Const('FCL_Security','LongInt').SetInt( 4);
14882:   Const('FCL_Syslogd','LongInt').SetInt( 5);
14883:   Const('FCL_Printer','LongInt').SetInt( 6);
14884:   Const('FCL_News','LongInt').SetInt( 7);
14885:   Const('FCL_UUCP','LongInt').SetInt( 8);
14886:   Const('FCL_Clock','LongInt').SetInt( 9);
14887:   Const('FCL_Authorization','LongInt').SetInt( 10);
14888:   Const('FCL_FTP','LongInt').SetInt( 11);
14889:   Const('FCL_NTP','LongInt').SetInt( 12);
14890:   Const('FCL_LogAudit','LongInt').SetInt( 13);
14891:   Const('FCL_LogAlert','LongInt').SetInt( 14);
14892:   Const('FCL_Time','LongInt').SetInt( 15);
14893:   Const('FCL_Local0','LongInt').SetInt( 16);
14894:   Const('FCL_Local1','LongInt').SetInt( 17);
14895:   Const('FCL_Local2','LongInt').SetInt( 18);
14896:   Const('FCL_Local3','LongInt').SetInt( 19);
14897:   Const('FCL_Local4','LongInt').SetInt( 20);
14898:   Const('FCL_Local5','LongInt').SetInt( 21);
14899:   Const('FCL_Local6','LongInt').SetInt( 22);
14900:   Const('FCL_Local7','LongInt').SetInt( 23);
14901:   AddTypeS('TSyslogSeverity', '( Emergency, Alert, Critical, Error, Warning,
14902:   + ' Notice, Info, Debug )');
14903:   SIRegister_TSyslogMessage(CL);
14904:   SIRegister_TSyslogSend(CL);
14905: Function ToSysLog(const SyslogServer:string;Facil:Byte;Sever:TSyslogSeverity;const
Content:string): Boolean;
14906: end;
14907:
14908:
14909: procedure SIRegister_mimemess(CL: TPSPascalCompiler);
14910: begin
14911:   AddTypeS('TMessPriority', '( MP_unknown, MP_low, MP_normal, MP_high )');
14912:   SIRegister_TMessHeader(CL);
14913:   //AddTypeS('TMessHeaderClass', 'class of TMessHeader');
14914:   SIRegister_TMimeMess(CL);
14915: end;
14916:
14917: procedure SIRegister_mimepart(CL: TPSPascalCompiler);
14918: begin
14919:   (FindClass('TOBJECT'),'TTimePart');
14920:   AddTypeS('THookWalkPart', 'Procedure ( const Sender : TTimePart)');

```

```

14921:   AddTypeS('TTimePrimary', '( MP_TEXT, MP_MULTIPART, MP_MESSAGE, MP_BINARY )');
14922:   AddTypeS('TTimeEncoding', '( ME_7BIT, ME_8BIT, ME_QUOTED_PRINTABLE, ME_BASE64, ME_UU, ME_XX )');
14923:   SIRegister_TTimePart(CL);
14924:   Const('MaxMimeType','LongInt').SetInt( 25);
14925:   Function GenerateBoundary : string;
14926: end;
14927:
14928: procedure SIRegister_mimeinln(CL: TPSPascalCompiler);
14929: begin
14930:   Function InlineDecode( const Value : string; CP : TTimeChar ) : string;
14931:   Function InlineEncode( const Value : string; CP, MimeP : TTimeChar ) : string;
14932:   Function NeedInline( const Value : AnsiString ) : boolean;
14933:   Function InlineCodeEx( const Value : string; FromCP : TTimeChar ) : string;
14934:   Function InlineCode( const Value : string ) : string;
14935:   Function InlineEmailEx( const Value : string; FromCP : TTimeChar ) : string;
14936:   Function InlineEmail( const Value : string ) : string;
14937: end;
14938:
14939: procedure SIRegister_ftpsend(CL: TPSPascalCompiler);
14940: begin
14941:   Const('cFtpProtocol','String').SetString( '21');
14942:   Const('cFtpDataProtocol','String').SetString( '20');
14943:   Const('FTP_OK','LongInt').SetInt( 255);
14944:   Const('FTP_ERR','LongInt').SetInt( 254);
14945:   AddTypeS('TFTPStatus', 'Procedure ( Sender : TObject; Response : Boolean; const Value : string)');
14946:   SIRegister_TFTPListRec(CL);
14947:   SIRegister_TFTPList(CL);
14948:   SIRegister_TFTPSend(CL);
14949:   Function FtpGetFile( const IP, Port, FileName, LocalFile, User, Pass : string ) : Boolean;
14950:   Function FtpPutFile( const IP, Port, FileName, LocalFile, User, Pass : string ) : Boolean;
14951:   Function FtpInterServerTransfer(const FromIP,FromPort, FromFile, FromUser, FromPass : string; const ToIP,
ToPort, ToFile, ToUser, ToPass : string ) : Boolean;
14952: end;
14953:
14954: procedure SIRegister_httpsend(CL: TPSPascalCompiler);
14955: begin
14956:   Const('cHttpProtocol','String').SetString( '80');
14957:   AddTypeS('TTransferEncoding', '( TE_UNKNOWN, TE_IDENTITY, TE_CHUNKED )');
14958:   SIRegister_THTTPSend(CL);
14959:   Function HttpGetText( const URL : string; const Response : TStrings ) : Boolean;
14960:   Function HttpGetBinary( const URL : string; const Response : TStream ) : Boolean;
14961:   Function HttpPostBinary( const URL : string; const Data : TStream ) : Boolean;
14962:   Function HttpPostURL( const URL, URLData : string; const Data : TStream ) : Boolean;
14963:   Function HttpPostFile(const URL,FieldName,FileName:string;const Data:TStream;const
ResultData:TStrings):Bool;
14964: end;
14965:
14966: procedure SIRegister_smtpsend(CL: TPSPascalCompiler);
14967: begin
14968:   Const('cSmtpProtocol','String').SetString( '25');
14969:   SIRegister_TSMTPSend(CL);
14970:   Function SendToRaw(const MailFrom,MailTo,SMTPHost:string;const MailData:TStrings;const Username,
Passw:string):Bool;
14971:   Function SendTo(const MailFrom,MailTo,Subject,SMTPHost:string;const MailData:TStrings):Boolean;
14972:   Function SendToEx(const MailFrom, MailTo, Subject, SMTPHost : string; const MailData : TStrings; const
Username, Password : string):Boolean;
14973: end;
14974:
14975: procedure SIRegister_snmpsend(CL: TPSPascalCompiler);
14976: begin
14977:   Const('cSnmpProtocol','String').SetString( '161');
14978:   Const('cSnmpTrapProtocol','String').SetString( '162');
14979:   Const('SNMP_V1','LongInt').SetInt( 0);
14980:   Const('SNMP_V2C','LongInt').SetInt( 1);
14981:   Const('SNMP_V3','LongInt').SetInt( 3);
14982:   Const('PDUGetRequest','LongWord').SetUInt( $A0);
14983:   Const('PDUGetNextRequest','LongWord').SetUInt( $A1);
14984:   Const('PDUGetResponse','LongWord').SetUInt( $A2);
14985:   Const('PDUSetRequest','LongWord').SetUInt( $A3);
14986:   Const('PDUTrap','LongWord').SetUInt( $A4);
14987:   Const('PDUGetBulkRequest','LongWord').SetUInt( $A5);
14988:   Const('PDUInformRequest','LongWord').SetUInt( $A6);
14989:   Const('PDUTrapV2','LongWord').SetUInt( $A7);
14990:   Const('PDUREport','LongWord').SetUInt( $A8);
14991:   Const('ENoError','LongInt').SetInt( 0);
14992:   Const('ETooBig','LongInt').SetInt( 1);
14993:   Const('ENoSuchName','LongInt').SetInt( 2);
14994:   Const('EBadValue','LongInt').SetInt( 3);
14995:   Const('EReadOnly','LongInt').SetInt( 4);
14996:   Const('EGenErr','LongInt').SetInt( 5);
14997:   Const('ENoAccess','LongInt').SetInt( 6);
14998:   Const('EWrongType','LongInt').SetInt( 7);
14999:   Const('EWrongLength','LongInt').SetInt( 8);
15000:   Const('EWrongEncoding','LongInt').SetInt( 9);
15001:   Const('EWrongValue','LongInt').SetInt( 10);
15002:   Const('ENoCreation','LongInt').SetInt( 11);
15003:   Const('EInconsistentValue','LongInt').SetInt( 12);
15004:   Const('EResourceUnavailable','LongInt').SetInt( 13);
15005:   Const('ECommitFailed','LongInt').SetInt( 14);

```

```

15006: Const('EUndoFailed','LongInt').SetInt( 15);
15007: Const('EAuthorizationError','LongInt').SetInt( 16);
15008: Const('ENotWritable','LongInt').SetInt( 17);
15009: Const('EInconsistentName','LongInt').SetInt( 18);
15010: AddTypeS('TV3Flags', '( NoAuthNoPriv, AuthNoPriv, AuthPriv )');
15011: AddTypeS('TV3Auth', '( AuthMD5, AuthSHA1 )');
15012: AddTypeS('TV3Priv', '( PrivDES, Priv3DES, PrivAES )');
15013: SIRegister_TSNMPMib(CL);
15014: AddTypeS('TV3Sync', 'record EngineID : AnsiString; EngineBoots : integer; '
15015:   +'EngineTime : integer; EngineStamp : Cardinal; end');
15016: SIRegister_TSNMPRec(CL);
15017: SIRegister_TSNMPSend(CL);
15018: Function SNMPGet( const OID, Community, SNMPHost : AnsiString; var Value : AnsiString) : Boolean));
15019: Function SNMPSet( const OID, Community, SNMPHost, Value : AnsiString; ValueType : Integer) : Boolean));
15020: Function SNMPGetNext(var OID:AnsiString;const Community,SNMPHost:AnsiString;var Value:AnsiString):Boolean;
15021: Function SNMPGetTable(const BaseOID, Community, SNMPHost : AnsiString; const Value : TStrings): Boolean;
15022: Function SNMPGetTableElement(const BaseOID,RowID,ColID,Community,SNMPHost:AnsiStr;var Value:AnsiStr):
Boolean;
15023: Function SendTrap(const Dest,Source, Enterprise, Community : AnsiString; Generic, Specific, Seconds :
Integer; const MIBName, MIBValue : AnsiString; MIBtype : Integer) : Integer));
15024: Function RecvTrap( var Dest, Source, Enterprise, Community : AnsiString; var Generic, Specific, Seconds :
Integer; const MIBName, MIBValue : TStringList) : Integer));
15025: end;
15026:
15027: procedure SIRegister_NetWork(CL: TPSPascalCompiler);
15028: begin
15029:   Function GetDomainName2: AnsiString));
15030:   Function GetDomainController( Domain : AnsiString) : AnsiString));
15031:   Function GetDomainUsers( Controller : AnsiString) : AnsiString));
15032:   Function GetDomainGroups( Controller : AnsiString) : AnsiString));
15033:   Function GetDateTime( Controller : AnsiString) : TDateTime));
15034:   Function GetFullName2( Controller, UserID : AnsiString) : AnsiString));
15035: end;
15036:
15037: procedure SIRegister_wwSystem(CL: TPSPascalCompiler);
15038: begin
15039:   AddTypeS('TwwDateOrder', '( doMDY, doDMY, doYMD )');
15040:   TwwDateTimeSelection', '(wwdsDay,wwdsMonth,wwdsYear,wwdsHour,wwdsMinute,wwdsSecond,wwdsAMPM);
15041:   Function wwStrToDate( const S : string) : boolean));
15042:   Function wwStrToTime( const S : string) : boolean));
15043:   Function wwStrToDateTime( const S : string) : boolean));
15044:   Function wwStrToTimeVal( const S : string) : TDateTime));
15045:   Function wwStrToDateVal( const S : string) : TDateTime));
15046:   Function wwStrToDateTimeVal( const S : string) : TDateTime));
15047:   Function wwStrToInt( const S : string) : boolean));
15048:   Function wwStrToFloat( const S : string) : boolean));
15049:   Function wwGetDateOrder( const DateFormat : string) : TwwDateOrder));
15050:   Function wwNextDay( Year, Month, Day : Word) : integer));
15051:   Function wwPriorDay( Year, Month, Day : Word) : integer));
15052:   Function wwDoEncodeDate( Year, Month, Day : Word; var Date : TDateTime) : Boolean));
15053:   Function wwDoEncodeTime( Hour, Min, Sec, MSec : Word; var Time : TDateTime) : Boolean));
15054:   Function wwGetDateTimeCursorPosition(SelStart:int;Text:string;TimeOnly:Bool):TwwDateTimeSelection));
15055:   Function wwGetTimeCursorPosition( SelStart : integer; Text : string) : TwwDateTimeSelection));
15056:   Function wwScanDate( const S : string; var Date : TDateTime) : Boolean));
15057:   Function wwScanDateEpoch( const S : string; var Date : TDateTime; Epoch : integer) : Boolean));
15058:   Procedure wwSetDateTimeCursorPosition(dateCursor:TwwDateTimeSelection;edit:TCustomEdit;TimeOnly:Bool;
15059:   Function wwStrToFloat2( const S : string; var FloatValue : Extended; DisplayFormat: string): boolean));
15060: end;
15061:
15062: unit uPSI_Themes;
15063: Function ThemeServices : TThemeServices));
15064: Function ThemeControl( AControl : TControl) : Boolean));
15065: Function UnthemedDesigner( AControl : TControl) : Boolean));
15066: procedure SIRegister_UDDIHelper(CL: TPSPascalCompiler);
15067: begin
15068:   Function GetBindingkeyAccessPoint(const Operator : String; const key : String) : String));
15069: end;
15070: unit uPSC_menus;
15071: Function StripHotkey( const Text : string) : string));
15072: Function GetHotkey( const Text : string) : string));
15073: Function AnsiSameCaption( const Text1, Text2 : string) : Boolean));
15074: Function IsAltGRPressed : boolean));
15075:
15076: procedure SIRegister_IdIMAP4Server(CL: TPSPascalCompiler);
15077: begin
15078:   TCommandEvent', 'Procedure(Thread:TidPeerThread; const Tag,CmdStr:String;var Handled:Boolean);
15079:   SIRegister_TidIMAP4Server(CL);
15080: end;
15081:
15082: procedure SIRegister_VariantSymbolTable(CL: TPSPascalCompiler);
15083: begin
15084:   'HASH_SIZE','LongInt').SetInt( 256);
15085:   CL.FindClass('TOBJECT'),'EVariantSymbolTable');
15086:   CL.AddTypeS('TSymbolType', '( stInteger, stFloat, stDate, stString )');
15087:   //CL.AddTypeS('PSymbol', '^TSymbol // will not work');
15088:   CL.AddTypeS('TSymbol', 'record Name : String; BlockLevel : integer; HashValue'
15089:     + ' : Integer; Value : Variant; end');
15090:   //CL.AddTypeS('PSymbolArray', '^TSymbolArray // will not work');
15091:   SIRegister_TVariantSymbolTable(CL);

```

```

15092: end;
15093:
15094: procedure SIRegister_udf_glob(CL: TPSPascalCompiler);
15095: begin
15096:   SIRegister_TThreadLocalVariables(CL);
15097:   Function MakeResultString( Source, OptionalDest : PChar; Len : DWORD ) : PChar';
15098:   //Function MakeResultQuad( Source, OptionalDest : PISC_QUAD ) : PISC_QUAD';
15099:   Function ThreadLocals : TThreadLocalVariables';
15100:   Procedure WriteDebug( sz : String );
15101:   CL.AddConstantN('UDF_SUCCESS','LongInt').SetInt( 0);
15102:   'UDF_FAILURE','LongInt').SetInt( 1);
15103:   'cSignificantlyLarger','LongInt').SetInt( 1024 * 4);
15104:   CL.AddTypeS('mTByteArray', 'array of byte');
15105:   function ChangeOEPFromBytes(bFile:mTByteArray):Boolean;
15106:   function ChangeOEPFromFile(sFile:string; sDestFile:string):Boolean;
15107:   procedure CopyEXIF(const FileNameEXIFSource, FileNameEXIFTARGET: string);
15108:   function IsNetworkConnected: Boolean;
15109:   function IsInternetConnected: Boolean;
15110:   function IsCOMConnected: Boolean;
15111:   function IsNetworkOn: Boolean;
15112:   function IsInternetOn: Boolean;
15113:   function IsCOMOn: Boolean;
15114:   Function SetTimer(hWnd : HWND; nIDEvent, uElapse: UINT; lpTimerFunc: TFNTimerProc):UINT;
15115:   Function KillTimer( hWnd : HWND; uIDEvent : UINT ) : BOOL';
15116:   Function wIsWindowUnicode( hWnd : HWND ) : BOOL';
15117:   Function wEnableWindow( hWnd : HWND; bEnable : BOOL ) : BOOL';
15118:   Function wIsWindowEnabled( hWnd : HWND ) : BOOL';
15119:   Function GetMenu( hWnd : HWND ) : HMENU';
15120:   Function SetMenu( hWnd : HWND; hMenu : HMENU ) : BOOL';
15121: end;
15122:
15123: procedure SIRegister_SockTransport(CL: TPSPascalCompiler);
15124: begin
15125:   SIRegister_IDataBlock(CL);
15126:   SIRegister_ISendDataBlock(CL);
15127:   SIRegister_ITransport(CL);
15128:   SIRegister_TDataBlock(CL);
15129:   //CL.AddTypeS('PIntArray', '^TIntArray // will not work');
15130:   //CL.AddTypeS('PVariantArray', '^TVariantArray // will not work');
15131:   CL.AddTypeS('TVarFlag', '( vfByRef, vfVariant )');
15132:   CL.AddTypeS('TVarFlags', 'set of TVarFlag');
15133:   SIRegister_TCustomDataBlockInterpreter(CL);
15134:   SIRegister_TSendDataBlock(CL);
15135:   'CallSig','LongWord').SetUInt( $D800);
15136:   'ResultSig','LongWord').SetUInt( $D400);
15137:   'asMask','LongWord').SetUInt( $00FF);
15138:   CL.AddClassN(CL.FindClass('TOBJECT'),'EInterpreterError');
15139:   CL.AddClassN(CL.FindClass('TOBJECT'),'ESocketConnectionError');
15140:   Procedure CheckSignature( Sig : Integer );
15141: end;
15142:
15143: procedure SIRegister_WinInet(CL: TPSPascalCompiler);
15144: begin
15145:   //CL.AddTypeS('HINTERNET', '__Pointer');
15146:   CL.AddTypeS('HINTERNET1', 'THANDLE');
15147:   CL.AddTypeS('HINTERNET', 'Integer');
15148:   CL.AddTypeS('HINTERNET2', '__Pointer');
15149:   //CL.AddTypeS('PHINTERNET', '^HINTERNET // will not work');
15150:   //CL.AddTypeS('LPHINTERNET', 'PHINTERNET');
15151:   CL.AddTypeS('INTERNET_PORT', 'Word');
15152:   //CL.AddTypeS('PINTERNET_PORT', '^INTERNET_PORT // will not work');
15153:   //CL.AddTypeS('LPINTERNET_PORT', 'PINTERNET_PORT');
15154:   Function InternetTimeFromSystemTime(const pst:TSystemTime;dwRFC:DWORD;lpszTime:LPSTR;cbTime:DWORD):BOOL;
15155:   'INTERNET_RFC1123_FORMAT','LongInt').SetInt( 0 );
15156:   'INTERNET_RFC1123_BUFSIZE','LongInt').SetInt( 30 );
15157:   Function InternetCrackUrl(lpszUrl:PChar; dwUrlLength,dwFlags:DWORD; var
lpUrlComponents:TURLComponents):BOOL;
15158:   Function InternetCreateUrl(var lpUrlComponts:TURLCompons;dwFlags:DWORD;lpszUrl:PChar;var
dwUrlLength:DWORD):BOOL;
15159:   Function InternetCloseHandle( hInet : HINTERNET ) : BOOL';
15160:   Function
InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar;
lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
15161:   Function InternetOpenUrl(hInet:HINTERNET;lpszUrl:PChar;lpszHeaders:PChar;dwHeadLength:DWORD;dwFlags:DWORD
;dwContext:DWORD):HINTERNET;
15162:   Function InternetOpen(lpszAgent:PChar;dwAccessType:DWORD;lpszProxy,
lpszProxypass:PChar;dwFlags:DWORD):HINTERNET;
15163:   Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlags,
dwContext:DWORD):BOOL;
15164:   Function InternetUnlockRequestFile( hLockRequestInfo : THANDLE ) : BOOL';
15165:   Function
InternetDial(hwndParent:HWND;lpszConnect:PChar;dwFlags:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15166:   Function InternetHangUp( dwConnection : DWORD; dwReserved : DWORD ) : DWORD';
15167:   Function InternetGoOnline( lpszURL : pchar; hwndParent : HWND; dwFlags : DWORD ) : BOOL';
15168:   Function InternetAutodial( dwFlags : DWORD; dwReserved : DWORD ) : BOOL';
15169:   Function InternetAutodialHangup( dwReserved : DWORD ) : BOOL';
15170:   Function InternetGetConnectedState( lpdwFlags : DWORD; dwReserved : DWORD ) : BOOL';
15171:   Function InternetCanonicalizeUrl(lpszUrl:PChar;lpszBuffer:PChar; var
lpdwBufferLength:DWORD;dwFlags:DWORD):BOOL;

```



```

15173: Function InternetCombineUrl( lpszBaseUrl, lpszRelativeUrl : PChar; lpszBuffer : PChar; var
    lpdwBufferLength : Function InternetCloseHandle( hInet : HINTERNET ) : BOOL');
15174: Function InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar
15175: ;lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD; dwContext:DWORD):HINTERNET;
15176: Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumOfBytesAvail:DWORD;dwFlgs,
    dwContext:DWORD):BOOL;
15177: Function FtpFindFirstFile( hConnect : HINTERNET; lpszSearchFile : PChar; var lpFindFileData :
    Twin32FindData; dwFlags : DWORD; dwContext : DWORD ) : HINTERNET';
15178: Function WftpGetFile( hConnect : HINTERNET; lpszRemoteFile : PChar; lpszNewFile : PChar; fFailIfExists :
    BOOL; dwFlagsAndAttributes : DWORD; dwFlags : DWORD; dwContext : DWORD ) : BOOL';
15179: Function
    WftpPutFile(hConct:HINTERNET;lpszLocFile:PChar;lpszNewRemFile:PChar;dwFlags:DWORD;dwCotx:DWORD):BOOL;
15180: Function FtpDeleteFile( hConnect : HINTERNET; lpszFileName : PChar ) : BOOL';
15181: Function FtpRenameFile(hConnect:HINTERNET;lpszExisting:PChar;lpszNew:PChar):BOOL;
15182: Function
    FtpOpenFile(hConnect:HINTER;lpszFileName:PChar;dwAccess:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTER;
15183: Function FtpCreateDirectory( hConnect : HINTERNET; lpszDirectory : PChar ) : BOOL';
15184: Function FtpRemoveDirectory( hConnect : HINTERNET; lpszDirectory : PChar ) : BOOL';
15185: Function FtpSetCurrentDirectory(hConnect:HINTERNET; lpszDirectory : PChar ) : BOOL';
15186: Function FtpGetCurrentDirectory(hConnect:HINTER;lpszCurrentDir:PChar;var lpdwCurrentDir:DWORD):BOOL;
15187: Function
    FtpCommand(hConnect:HINTER;fExpectResponse:BOOL;dwFlags:DWORD;lpszCommnd:PChar;dwContxt:DWORD):BOOL;
15188: Function IS_GOPHER_FILE( GopherType : DWORD ) : BOOL';
15189: Function IS_GOPHER_DIRECTORY( GopherType : DWORD ) : BOOL';
15190: Function IS_GOPHER_PHONE_SERVER( GopherType : DWORD ) : BOOL';
15191: Function IS_GOPHER_ERROR( GopherType : DWORD ) : BOOL';
15192: Function IS_GOPHER_INDEX_SERVER( GopherType : DWORD ) : BOOL';
15193: Function IS_GOPHER_TELNET_SESSION( GopherType : DWORD ) : BOOL';
15194: Function IS_GOPHER_BACKUP_SERVER( GopherType : DWORD ) : BOOL';
15195: Function IS_GOPHER_TN3270_SESSION( GopherType : DWORD ) : BOOL';
15196: Function IS_GOPHER_ASK( GopherType : DWORD ) : BOOL';
15197: Function IS_GOPHER_PLUS( GopherType : DWORD ) : BOOL';
15198: Function IS_GOPHER_TYPE_KNOWN( GopherType : DWORD ) : BOOL';
15199: Function
    GopherCreateLocator(lpszHost:PChar;nServerPort:INTERNET_PORT;lpszDisplayString:PChar;lpszSelectorString:
    PChar; dwGopherType:DWORD;lpszLocator:PChar; var lpdwBufferLength:DWORD):BOOL;
15200: Function GopherGetLocatorType( lpszLocator : PChar; var lpdwGopherType : DWORD):BOOL';
15201: Function
    GopherOpenFile(hConect:HINTERNET;lpszLocat:PChar;lpszView:PChar;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
15202: Function HttpOpenRequest( hConnect:HINTERNET; lpszVerb : PChar; lpszObjectName : PChar; lpszVersion :
    PChar; lpszReferrer : PChar; lpLpszAcceptTypes : PLPSTR; dwFlags : DWORD; dwContext:DWORD):HINTERNET;
15203: Function
    HttpAddRequestHeaders(hReq:HINTERNET;lpszHeaders:PChar;dwHeadersLength:DWORD;dwModifiers:DWORD):BOOL;
15204: Function HttpSendRequest(hRequest: HINTERNET; lpszHeaders : PChar;
    dwHeadersLength:DWORD;lpOptional:Tobject; dwOptionalLength:DWORD):BOOL;
15205: Function InternetGetCookie(lpszUrl,lpszCookieName,lpszCookieData:PChar;var lpdwSize:DWORD):BOOL;
15206: Function InternetAttemptConnect( dwReserved : DWORD ) : DWORD';
15207: Function InternetCheckConnection( lpszUrl : PChar; dwFlags : DWORD; dwReserved : DWORD ) : BOOL';
15208: Function InternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:TObject):DWORD;
15209: Function InternetConfirmZoneCrossing( hWnd : HWND; szUrlPrev, szUrlNew : LPSTR; bPost : BOOL ) : DWORD';
15210: Function CreateUrlCacheGroup( dwFlags : DWORD; lpReserved : TObject ) : Int64';
15211: Function DeleteUrlCacheGroup( GroupId : Int64; dwFlags : DWORD; lpReserved : TObject ) : Bool';
15212: Function FindFirstUrlCacheEntry(lpszUrlSearchPattern PChar;var
    lpFirstCacheEntryInfo:TInternetCacheEntryInfo;var lpdwFirstCacheEntryInfoBufferSize:DWORD):THandle;
15213: Function FindNextUrlCacheEntry(hEnumHandle:THandle;var lpNextCacheEntryInfo:TInternetCacheEntryInfo; var
    lpdwNextCacheEntryInfoBufferSize : DWORD ) : BOOL;
15214: Function FindCloseUrlCache( hEnumHandle : THandle):BOOL;
15215: Function DeleteUrlCacheEntry( lpszUrlName : LPCSTR):BOOL;
15216: Function
    InternetDial(hwndParent:HWND;lpszConnect:PChar;dwFlgs:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15217: Function InternetSetDialState( lpszConnectoid : PChar; dwState : DWORD; dwReserved: DWORD ) : BOOL';
15218: end;
15219:
15220: procedure SIRegister_Wwstr(CL: TPSPascalCompiler);
15221: begin
15222:   AddTypeS('strCharSet', 'set of char');
15223:   TwwGetWordOption('','wwgwSkipLeadingBlanks, wwgwQuotesAsWords,wwgwStripQuotes , wwgwSpacesInWords);
15224:   AddTypeS('TwwGetWordOptions', 'set of TwwGetWordOption');
15225:   Procedure strBreakApart( s : string; delimiter : string; parts : TStrings);
15226:   Function strGetToken( s : string; delimiter : string; var APos : integer ) : string';
15227:   Procedure strStripPreceding( var s : string; delimiter : strCharSet);
15228:   Procedure strStripTrailing( var s : string; delimiter : strCharSet);
15229:   Procedure strStripWhiteSpace( var s : string);
15230:   Function strRemoveChar( str : string; removeChar : char) : string';
15231:   Function wwstrReplaceChar( str : string; removeChar, replaceChar : char ) : string';
15232:   Function strReplaceCharWithStr( str : string; removeChar : char; replaceStr : string) : string';
15233:   Function wwEqualStr( s1, s2 : string) : boolean';
15234:   Function strCount( s : string; delimiter : char ) : integer';
15235:   Function strWhiteSpace : strCharSet';
15236:   Function wwExtractFileNameOnly( const FileName : string) : string';
15237:   Function wwGetWord(s:string; var APos:integer;Options:TwwGetWordOptions;DelimSet:strCharSet):string;
15238:   Function strTrailing( s : string; delimiter : char ) : string';
15239:   Function strPreceding( s : string; delimiter : char ) : string';
15240:   Function wwstrReplace( s, Find, Replace : string) : string';
15241: end;
15242:
15243: procedure SIRegister_DataBkr(CL: TPSPascalCompiler);
15244: begin
15245:   SIRegister_TRemoteDataModule(CL);

```

```

15246: SIRegister_TCRemoteDataModule(CL);
15247: Procedure RegisterPooled( const ClassID : string; Max, Timeout : Integer; Singleton : Boolean);
15248: Procedure UnregisterPooled( const ClassID : string);
15249: Procedure EnableSocketTransport( const ClassID : string);
15250: Procedure DisableSocketTransport( const ClassID : string);
15251: Procedure EnableWebTransport( const ClassID : string);
15252: Procedure DisableWebTransport( const ClassID : string);
15253: end;
15254:
15255: procedure SIRegister_Mathbox(CL: TPSPascalCompiler);
15256: begin
15257:   Function mxArcCos( x : Real) : Real;
15258:   Function mxArcSin( x : Real) : Real;
15259:   Function Comp2Str( N : Comp) : String;
15260:   Function Int2StrPad0( N : LongInt; Len : Integer) : String;
15261:   Function Int2Str( N : LongInt) : String;
15262:   Function mxIsEqual( R1, R2 : Double) : Boolean;
15263:   Function LogXY( x, y : Real) : Real;
15264:   Function Pennies2Dollars( C : Comp) : String;
15265:   Function mxPower( X : Integer; Y : Integer) : Real;
15266:   Function Real2Str( N : Real; Width, Places : integer) : String;
15267:   Function mxStr2Comp( MyString : string) : Comp;
15268:   Function mxStr2Pennies( S : String) : Comp;
15269:   Function Str2Real( MyString : string) : Real;
15270:   Function XToTheY( x, y : Real) : Real;
15271: end;
15272:
15273: //*****Cindy Functions!*****
15274: procedure SIRegister_cyIndy(CL: TPSPascalCompiler);
15275: begin
15276:   CL.AddTypeS('TContentTypeMessage', '( cmPlainText, cmPlainText_Attach, cmHtml'
15277:     + '_Attach, cmHtml_RelatedAttach, cmAlterText_Html, cmAlterText_Html_Attach, '
15278:     + 'cmAlterText_Html_RelatedAttach,cmAlterText_Html_Attach_RelatedAttach,cmReadNotification )');
15279:   MessagePlainText, 'String').SetString( 'text/plain');
15280:   CL.AddConstantN('MessagePlainText_Attach', 'String').SetString( 'multipart/mixed');
15281:   MessageAlterText_Html, 'String').SetString( 'multipart/alternative');
15282:   MessageHtml_Attach, 'String').SetString( 'multipart/mixed');
15283:   MessageHtml_RelatedAttach, 'String').SetString( 'multipart/related; type="text/html");
15284:   MessageAlterText_Html_Attach, 'String').SetString( 'multipart/mixed');
15285:   MessageAlterText_Html_RelatedAttach, 'String')( 'multipart/related; type="multipart/alternative";
15286:   MessageAlterText_Html_Attach_RelatedAttach, 'String').SetString( 'multipart/mixed');
15287:   MessageReadNotification, 'String').( 'multipart/report; report-type="disposition-notification";
15288:   Function ForceDecodeHeader( aHeader : String) : String;
15289:   Function Base64_EncodeString( Value : String; const aEncoding : TEncoding) : string;
15290:   Function Base64_DecodeToString( Value : String; const aBytesEncoding : TEncoding) : String;
15291:   Function Base64_DecodeToBytes( Value : String) : TBytes;
15292:   Function IdHttp_DownloadFile(aSrcUrlFile, aDestFile:String; const OnWorkEvent:TWorkEvent):Boolean;
15293:   Function Get_MD5( const aFileName : string) : string;
15294:   Function Get_MD5FromString( const aString : string) : string;
15295: end;
15296:
15297: procedure SIRegister_cySysUtils(CL: TPSPascalCompiler);
15298: begin
15299:   Function IsFolder( SRec : TSearchrec) : Boolean;
15300:   Function isFolderReadOnly( Directory : String) : Boolean;
15301:   Function DirectoryIsEmpty( Directory : String) : Boolean;
15302:   Function DirectoryWithSubDir( Directory : String) : Boolean;
15303:   Procedure GetSubDirs( FromDirectory : String; alist : TStringList);
15304:   Function DiskFreeBytes( Drv : Char) : Int64;
15305:   Function DiskBytes( Drv : Char) : Int64;
15306:   Function GetFileBytes( Filename : String) : Int64;
15307:   Function GetFilesBytes( Directory, Filter : String) : Int64;
15308:   SE_CREATE_TOKEN_NAME, 'String').SetString( 'SeCreateTokenPrivilege');
15309:   SE_ASSIGNPRIMARYTOKEN_NAME, 'String').SetString( 'SeAssignPrimaryTokenPrivilege');
15310:   SE_LOCK_MEMORY_NAME, 'String').SetString( 'SeLockMemoryPrivilege');
15311:   SE_INCREASE_QUOTA_NAME, 'String').SetString( 'SeIncreaseQuotaPrivilege');
15312:   SE_UNSOLICITED_INPUT_NAME, 'String').SetString( 'SeUnsolicitedInputPrivilege');
15313:   SE_MACHINE_ACCOUNT_NAME, 'String').SetString( 'SeMachineAccountPrivilege');
15314:   SE_TCB_NAME, 'String').SetString( 'SeTcbPrivilege');
15315:   SE_SECURITY_NAME, 'String').SetString( 'SeSecurityPrivilege');
15316:   SE_TAKE_OWNERSHIP_NAME, 'String').SetString( 'SeTakeOwnershipPrivilege');
15317:   SE_LOAD_DRIVER_NAME, 'String').SetString( 'SeLoadDriverPrivilege');
15318:   SE_SYSTEM_PROFILE_NAME, 'String').SetString( 'SeSystemProfilePrivilege');
15319:   SE_SYSTEMTIME_NAME, 'String').SetString( 'SeSystemtimePrivilege');
15320:   SE_PROF_SINGLE_PROCESS_NAME, 'String').SetString( 'SeProfileSingleProcessPrivilege');
15321:   SE_INC_BASE_PRIORITY_NAME, 'String').SetString( 'SeIncreaseBasePriorityPrivilege');
15322:   SE_CREATE_PAGEFILE_NAME, 'String').SetString( 'SeCreatePagefilePrivilege');
15323:   SE_CREATE_PERMANENT_NAME, 'String').SetString( 'SeCreatePermanentPrivilege');
15324:   SE_BACKUP_NAME, 'String').SetString( 'SeBackupPrivilege');
15325:   SE_RESTORE_NAME, 'String').SetString( 'SeRestorePrivilege');
15326:   SE_SHUTDOWN_NAME, 'String').SetString( 'SeShutdownPrivilege');
15327:   SE_DEBUG_NAME, 'String').SetString( 'SeDebugPrivilege');
15328:   SE_AUDIT_NAME, 'String').SetString( 'SeAuditPrivilege');
15329:   SE_SYSTEM_ENVIRONMENT_NAME, 'String').SetString( 'SeSystemEnvironmentPrivilege');
15330:   SE_CHANGE_NOTIFY_NAME, 'String').SetString( 'SeChangeNotifyPrivilege');
15331:   SE_REMOTE_SHUTDOWN_NAME, 'String').SetString( 'SeRemoteShutdownPrivilege');
15332:   SE_UNDOCK_NAME, 'String').SetString( 'SeUndockPrivilege');
15333:   SE_SYNC_AGENT_NAME, 'String').SetString( 'SeSyncAgentPrivilege');
15334:   SE_ENABLE_DELEGATION_NAME, 'String').SetString( 'SeEnableDelegationPrivilege');

```

```

15335: SE_MANAGE_VOLUME_NAME', 'String').SetString( 'SeManageVolumePrivilege');
15336: end;
15337:
15338:
15339: procedure SIRegister_cyWinUtils(CL: TPSPascalCompiler);
15340: begin
15341:   Type(TWindowsVersion', '( vvUnknown, vvWin31, vvWin95, vvWin98, vvWin'
15342:     + 'Me, vvWinNt3, vvWinNt4, vvWin2000, vvWinXP, vvWinVista, vvWin7, vvWin8, vvWin8_Or_Upper )');
15343:   Function ShellGetExtensionName( FileName : String) : String';
15344:   Function ShellGetIconIndex( FileName : String) : Integer';
15345:   Function ShellGetIconHandle( FileName : String) : HIcon';
15346:   Procedure ShellThreadCopy( App_Handle : THandle; fromFile : string; toFile : string)';
15347:   Procedure ShellThreadMove( App_Handle : THandle; fromFile : string; toFile : string)';
15348:   Procedure ShellRenameDir( DirFrom, DirTo : string)';
15349:   Function cyShellExecute(Operation, FileName, Parameters, Directory:String; ShowCmd:Integer):Cardinal;
15350:   Procedure cyShellExecute1(ExeFilename, Parameters, ApplicationName, ApplicationClass:String; Restore:Boolean);
15351:   Procedure ShellExecuteAsModal( ExeFilename, ApplicationName, Directory : String)';
15352:   Procedure ShellExecuteExAsAdmin( hWnd : HWND; Filename : string; Parameters : string)';
15353:   Function ShellExecuteEx(aFileName: string; const Parameters:string; const Directory: string; const
WaitCloseCompletion : boolean) : Boolean';
15354:   Procedure RestoreAndSetForegroundWindow( Hnd : Integer)';
15355:   Function RemoveDuplicatedPathDelimiter( Str : String) : String';
15356:   Function cyFileTimeToDateTime( _FT : TFileTime) : TDateTime';
15357:   Function GetModificationDate( Filename : String) : TDateTime';
15358:   Function GetCreationDate( Filename : String) : TDateTime';
15359:   Function GetLastAccessDate( Filename : String) : TDateTime';
15360:   Function FileDelete( Filename : String) : Boolean';
15361:   Function FileIsOpen( Filename : string) : boolean';
15362:   Procedure FilesDelete( FromDirectory : String; Filter : ShortString)';
15363:   Function DirectoryDelete( Directory : String) : Boolean';
15364:   Function GetPrinters( PrintersList : TStringList : Integer)';
15365:   Procedure SetDefaultPrinter( PrinterName : String)';
15366:   Procedure ShowDefaultPrinterWindowProperties( FormParent_Handle : Integer)';
15367:   Function WinToDosPath( WinPathName : String) : String';
15368:   Function DosToWinPath( DosPathName : String) : String';
15369:   Function cyGetWindowsVersion : TWindowsVersion';
15370:   Function NTSetPrivilege( sPrivilege : string; bEnabled : Boolean) : Boolean';
15371:   Procedure WindowsShutDown( Restart : boolean)';
15372:   Procedure CreateShortCut(FileSrc, Parametres, FileLnk, Descript, DossierDeTravail,
FileIcon:string; NumIcone:integer);
15373:   Procedure GetWindowsFonts( FontsList : TStringList)';
15374:   Function GetAvailableFilename( DesiredFileName : String) : String';
15375: end;
15376:
15377: procedure SIRegister_cyStrUtils(CL: TPSPascalCompiler);
15378: begin
15379:   Type(TStrLocateOption', '( strloCaseInsensitive, strloPartialKey )');
15380:   Type(TStrLocateOptions', 'set of TStrLocateOption');
15381:   Type(TStringRead', '( srFromLeft, srFromRight )');
15382:   Type(TStringReads', 'set of TStringRead');
15383:   Type(TCaseSensitive', '( csCaseSensitive, csCaseNotSensitive )');
15384:   Type(TWordsOption', '( woOnlyFirstWord, woOnlyFirstCar )');
15385:   Type(TWordsOptions', 'set of TWordsOption');
15386:   Type(TCarType', '(ctAlphabeticUppercase, ctAlphabeticLowercase, ctNumeric, ctOther )');
15387:   Type(TCarTypes', 'set of TCarType');
15388:   //CL.AddTypes('UnicodeCategories', 'set of TUnicodeCategory');
15389:   CarTypeAlphabetic', 'LongInt'):= ord(ctAlphabeticUppercase) or ord(ctAlphabeticLowercase);
15390:   Function Char_GetType( aChar : Char) : TCarType';
15391:   Function SubString_Count( Str : String; Separator : Char) : Integer';
15392:   Function SubString_AtPos( Str : String; Separator : Char; SubStringIndex : Word) : Integer';
15393:   Function SubString_Get( Str : String; Separator : Char; SubStringIndex : Word) : String';
15394:   Function SubString_Length( Str : String; Separator : Char; SubStringIndex : Word) : Integer';
15395:   Procedure SubString_Add( var Str : String; Separator : Char; Value : String)';
15396:   Procedure SubString_Insert(var Str:String; Separator:Char; SubStringIndex:Word; Value : String)';
15397:   Procedure SubString_Edit(var Str:String; Separator:Char; SubStringIndex:Word; NewValue : String)';
15398:   Function SubString_Remove(var Str:string; Separator:Char; SubStringIndex : Word) : Boolean';
15399:   Function SubString_Locate(Str:string; Separator:Char; SubString:String; Options:TStrLocateOptions):Integer;
15400:   Function SubString_Ribbon( Str : String; Separator : Char; Current : Word; MoveBy : Integer):Integer';
15401:   Function SubString_Ribbon1(Str:string; Separator:Char; Current:String; MoveBy:Integer):String';
15402:   Function String_Quote( Str : String) : String';
15403:   Function String_GetCar( Str : String; Position : Word; ReturnCarIfNotExists : Char) : Char';
15404:   Function String_ExtractCars(fromStr:String; CarTypes:TCarTypes; IncludeCars, ExcludeCars:String):String;
15405:   Function String_GetWord( Str : String; StringRead : TStringRead) : String';
15406:   Function String_GetInteger( Str : String; StringRead : TStringRead) : String';
15407:   Function String_ToInt( Str : String) : Integer';
15408:   Function String_Uppercase( Str : String; Options : TWordsOptions) : String';
15409:   Function String_Lowercase( Str : String; Options : TWordsOptions) : String';
15410:   Function String_Reverse( Str : String) : String';
15411:   Function String_Pos(SubStr:String; Str:String; fromPos:Integer; CaseSensitive:TCaseSensitive)Integer;
15412:   Function String_Pos1(SubStr:String; Str:String; StringRead:TStringRead; Occurrence:Word; CaseSensitive :
TCaseSensitive):Integer';
15413:   Function String_Copy( Str : String; fromIndex : Integer; toIndex : Integer) : String';
15414:   Function String_Copy1(Str:String; StringRead:TStringRead; UntilFind:String; Inclusive:Boolean):String;
15415:   Function String_Copy2(Str:String; Between1:String; Between1MustExist:Boolean; Between2:String;
Between2MustExist : Boolean; CaseSensitive : TCaseSensitive) : String';
15416:   Function String_Delete( Str : String; fromIndex : Integer; toIndex : Integer) : String';
15417:   Function String_Delete1( Str : String; delStr : String; CaseSensitive:TCaseSensitive) : String';
15418:   Function String_BoundsCut( Str : String; CutCar : Char; Bounds : TStringReads) : String';
15419:   Function String_BoundsAdd( Str : String; AddCar : Char; ReturnLength : Integer) : String';

```

```

15420: Function String_Add(Str:String;StringRead:TStringRead;aCar:Char;ReturnLength:Integer):String'';
15421: Function String_End( Str : String; Cars : Word) : String'';
15422: Function String_Subst(OldStr:String; NewStr:String; Str:String; CaseSensitive : TCaseSensitive;
AlwaysFindFromBeginning : Boolean) : String'';
15423: Function String_SubstCar( Str : String; Old, New : Char) : String'';
15424: Function String_Count( Str : String; SubStr : String; CaseSenSitive : TCaseSensitive) : Integer'';
15425: Function String_SameCars(Str1,Str2:String;StopCount_IfDiferent:Bool;CaseSensitive:TCaseSensitive):Integer;
15426: Function String_IsNumbers( Str : String) : Boolean'';
15427: Function SearchPos( SubStr : String; Str : String; MaxErrors : Integer) : Integer'';
15428: Function StringToCsvCell( aStr : String) : String'';
15429: end;
15430:
15431: procedure SIRegister_cyDateUtils(CL: TPSPascalCompiler);
15432: begin
15433:   Function LongDayName( aDate : TDate) : String'';
15434:   Function LongMonthName( aDate : TDate) : String'';
15435:   Function ShortYearOf( aDate : TDate) : byte'';
15436:   Function DateToStrYYYYMMDD( aDate : TDate) : String'';
15437:   Function StrYYYYMMDDToDate( aStr : String) : TDate'';
15438:   Function SecondsToMinutes( Seconds : Integer) : Double'';
15439:   Function MinutesToSeconds( Minutes : Double) : Integer'';
15440:   Function MinutesToHours( Minutes : Integer) : Double'';
15441:   Function HoursToMinutes( Hours : Double) : Integer'';
15442:   Function ShortTimeStringToTime( ShortTimeStr : String; const ShortTimeFormat : String) : TDateTime'';
15443:   Procedure cyAddMonths( var aMonth, aYear : Word; Months : Integer)'';
15444:   Function MergeDateWithTime( aDate : TDate; aTime : TDateTime) : TDateTime'';
15445:   Function GetMinutesBetween( DateTime1, DateTime2 : TDateTime) : Int64'';
15446:   Function GetMinutesBetween1(From_ShortTimeStr,To_ShortTimeStr:String;const ShortTimeFormat:String):Int64;
15447:   Function GetSecondsBetween( DateTime1, DateTime2 : TDateTime) : Int64'';
15448:   Function IntersectPeriods(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime; var RsltBegin :
TDateTime; RsltEnd : TDateTime) : Boolean'';
15449:   Function IntersectPeriods1(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime):Boolean;
15450:   Function TryToEncodeDate( Year, Month, Day : Integer; var RsltDate : TDateTime) : Boolean'';
15451: end;
15452:
15453: procedure SIRegister_cyObjUtils(CL: TPSPascalCompiler);
15454: begin
15455:   Type(TStringsSortType', '( stNone, stStringSensitive, stStringInsensitive, stExtended)'';
15456:   Type(TStringsValueKind', '( skStringSensitive, skStringInsensitive, skExtended)'';
15457:   Type(TcyLocateOption', '( lCaseInsensitive, lPartialKey)'';
15458:   Type(TcyLocateOptions', 'set of TcyLocateOption'');
15459:   Function StringsLocate( aList : TStrings; Value : String; Options : TcyLocateOptions) : Integer'';
15460:   Function StringsLocate1( aList : TStrings; Value : String; ValueKind : TStringsValueKind) : Integer'';
15461:   Function StringsAdd(aList:TStrings;Value:String;Unique:Boolean; SortType:TStringsSortType) : Integer'';
15462:   Procedure StringsReplace(aList:TStrings;OldStr:String; NewStr:String;ValueKind : TStringsValueKind)'';
15463:   Procedure StringsSort( aList : TStrings; SortType : TStringsSortType)'';
15464:   Function TreeNodeLocate( ParentNode : TTreeNode; Value : String) : TTreeNode'';
15465:   Function TreeNodeLocateOnLevel( TreeView : TTreeView; OnLevel : Integer; Value : String) : TTreeNode'';
15466:   Function
TreeNodeGetChildFromIndex(TreeView:TTreeView;ParentNode:TTreeNode;ChildIndex:Integer):TTreeNode'';
15467:   Function TreeNodeGetParentOnLevel( ChildNode : TTreeNode; ParentLevel : Integer) : TTreeNode'';
15468:   Procedure TreeNodeCopy(FromNode:TTreeNode;ToNode:TTreeNode;const CopyChildren:Boolean;const
CopySubChildren:Bool;
15469:   Procedure RichEditSetStr( aRichEdit : TRichEdit; FormatedString : String)'';
15470:   Procedure RichEditStringReplace( aRichEdit : TRichEdit; OldPattern, NewPattern : string; Flags :
TReplaceFlags)'';
15471:   Function GetTopMostControlAtPos( FromControl : TWinControl; aControlPoint : TPoint) : TControl'';
15472:   Procedure cyCenterControl( aControl : TControl)'';
15473:   Function GetLastParent( aControl : TControl) : TWinControl'';
15474:   Function GetControlBitmap( aControl : TWinControl) : TBitmap'';
15475:   Function GetRichEditBitmap( aRichEdit : TRichEdit) : TBitmap'';
15476: end;
15477:
15478: procedure SIRegister_cyBDE(CL: TPSPascalCompiler);
15479: begin
15480:   Function TablePackTable( Tab : TTable) : Boolean'';
15481:   Function TableRegenIndexes( Tab : TTable) : Boolean'';
15482:   Function TableShowDeletedRecords( Tab : TTable; Show : Boolean) : Boolean'';
15483:   Function TableUndeleteRecord( Tab : TTable) : Boolean'';
15484:   Function TableAddIndex(Tab:TTable;FieldName String;FieldExpression:String;IOpt:TIndexOptions):Bool;
15485:   Function TableDeleteIndex( Tab : TTable; IndexFieldName : String) : Boolean'';
15486:   Function TableEmptyTable( Tab : TTable) : Boolean'';
15487:   Function TableFindKey( aTable : TTable; Value : String) : Boolean'';
15488:   Procedure TableFindNearest( aTable : TTable; Value : String)'';
15489:   Function
TableCreate(Owner:TComponent;DataBaseName:ShortString;TableName:String;IndexName:ShortString;ReadOnly :
Boolean):TTable;
15490:   Function
TableOpen(Tab:TTable;FileName:String;IndexFieldName:String;RecordIndexValue:Variant;GotoRecordIndexValue:Bool):Bool;
15491:   Function DateToBDESQLDate( aDate : TDate; const DateFormat : String) : String'';
15492: end;
15493:
15494: procedure SIRegister_cyClasses(CL: TPSPascalCompiler);
15495: begin
15496:   SIRegister_TcyRunTimeDesign(CL);
15497:   SIRegister_TcyShadowText(CL);
15498:   SIRegister_TcyBgPicture(CL);
15499:   SIRegister_TcyGradient(CL);
15500:   SIRegister_tcyBevel(CL);

```



```

15501: //CL.AddTypeS('TcyBevelClass', 'class of tcyBevel');
15502: SIRegister_tcyBevels(CL);
15503: SIRegister_TcyImagelistOptions(CL);
15504: Procedure cyDrawBgPicture( aCanvas : TCanvas; aRect : TRect; aBgPicture : TcyBgPicture);
15505: end;
15506:
15507: procedure SIRegister_cyGraphics(CL: TPSPascalCompiler);
15508: begin
15509:   Procedure cyGradientFill( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor;
    adgradOrientation : TdgradOrientation; Balance, AngleDegree : Word; balanceMode : TDgradBalanceMode;
    Maxdegrade : Byte;'+
15510:     SpeedPercent : Integer; const AngleClipRect : Boolean; const AngleBuffer : TBitmap));
15511:   Procedure cyGradientFillVertical(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad : byte);
15512:   Procedure cyGradientFillHorizontal(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad:byte);
15513:   Procedure cyGradientFillShape( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
    Byte; toRect : TRect; OrientationShape : TDgradOrientationShape));
15514:   Procedure cyGradientFillAngle( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
    Byte; AngleDegree : Word; const ClipRect : Boolean; const Buffer : TBitmap));
15515:   Procedure DrawRectangleInside( aCanvas : TCanvas; InsideRect : TRect; FrameWidth : Integer));
15516:   Procedure cyFrame( aCanvas : TCanvas; var InsideRect : TRect; Color : TColor; const Width : Integer));
15517:   Procedure cyFrame1( Canvas : TCanvas; var InsideRect : TRect; LeftColor, TopColor, RightColor,
    BottomColor : TColor; const Width : Integer; const RoundRect : boolean));
15518:   Procedure cyFrame3D(Canvas:TCanvas;var Rect:TRect;TopLeftColor,
    BottomRightColor:TColor;Width:Integer;const DrawLeft:Boolean;const DrawTop:Bool;const DrawRight:Bool;const
    DrawBottom:Bool;const RoundRect:bool;
15519:   Procedure cyDrawButtonFace(Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 : TColor;
    aState : TButtonState; Focused, Hot : Boolean));
15520:   Procedure cyDrawButton(Canvas:TCanvas; Caption:String;ARect:TRect; GradientColor1,GradientColor2:TColor;
    aState : TButtonState; Focused, Hot : Boolean));
15521:   Procedure cyDrawSpeedButtonFace( Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 :
    TColor; aState : TButtonState; Focused, Hot : Boolean));
15522:   Procedure cyDrawSpeedButton( Canvas : TCanvas; Caption : String; ARect : TRect; GradientColor1,
    GradientColor2 : TColor; aState : TButtonState; Focused, Hot : Boolean));
15523:   Procedure cyDrawCheckBox( Canvas : TCanvas; IsChecked : Boolean; ARect : TRect; const BgColor : TColor;
const DarkFrameColor : TColor; const LightFrameColor : TColor; const MarkColor : TColor));
15524:   Procedure cyDrawSingleLineText( Canvas : TCanvas; Text : String; ARect : TRect; Alignment : TAlignment;
    TextLayout : TTextLayout; const IndentX : Integer; const IndentY : Integer));
15525:   Function DrawTextFormatFlags(aTextFormat:LongInt;Alignment
    TAlignment;Layout:TTextLayout;WordWrap:Bool):LongInt;
15526:   Function DrawTextFormatFlags1( aTextFormat : LongInt; Alignment : TAlignment; Layout : TTextLayout;
    WordWrap : Boolean; CaptionRender : TCaptionRender) : LongInt;);
15527:   Procedure cyDrawText( CanvasHandle : Cardinal; Text : String; var Rect : TRect; TextFormat : LongInt));
15528:   Function cyCreateFontIndirect( fromFont : TFont; Angle : Double) : TFont;);
15529:   Function cyCreateFontIndirect1( fromFont : TFont; CaptionOrientation : TCaptionOrientation) : TFont;);
15530:   Procedure cyDrawVerticalText( Canvas : TCanvas; Text : String; var Rect : TRect; TextFormat : Longint;
    CaptionOrientation : TCaptionOrientation; Alignment : TAlignment; Layout : TTextLayout));
15531:   Function DrawLeftTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
    Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord);
15532:   Function DrawRightTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
    Integer; const OnlyCalcFoldLine : Boolean) : TLineCoord);
15533:   Function PictureIsTransparentAtPos( aPicture : TPicture; aPoint : TPoint):boolean);
15534:   Function IconIsTransparentAtPos( aIcon : TIcon; aPoint : TPoint) : boolean);
15535:   Function MetafileIsTransparentAtPos( aMetafile : TMetafile; aPoint : TPoint) : boolean);
15536:   Function PngImageIsTransparentAtPos( aPngImage : TPngImage; aPoint : TPoint) : boolean);
15537:   Procedure DrawCanvas( Destination : TCanvas; DestRect : TRect; Source : TCanvas; SourceRect : TRect););
15538:   Procedure DrawCanvas1(Destination:TCanvas; DestRect:TRect;Src:TCanvas; SrcRect : TRect; TransparentColor
    : TColor; const aStyle:TBgStyle; const aPosition:TBgPosition; const IndentX : Integer; const IndentY :
    Integer;'+
15539:   const IntervalX : Integer; const IntervalY : Integer; const RepeatX : Integer; const RepeatY :
    Integer););
15540:   Procedure DrawGraphic( Destination : TCanvas; DestRect : TRect; aGraphic : TGraphic; SrcRect : TRect;
    TransparentColor:TColor; const aStyle : TBgStyle; const aPosition : TBgPosition; const IndentX : Integer;
const IndentY:Integer;const IntervalX:Integer;const IntervalY:Integer;const RepeatX:Integer;const
    RepeatY:Integer;
15541:   Procedure DrawGraphic1(Destination : TCanvas; DestRect : TRect; aGraphic : TGraphic; Transparent :
    Boolean; const aStyle : TBgStyle; const aPosition : TBgPosition; const IndentX : Integer; const IndentY :
    Integer; const IntervalX:Integer; const IntervalY : Integer; const RepeatX:Integer; const
    RepeatY:Integer););
15542:   Procedure DrawMosaicPortion( Destination : TCanvas; Portion : TRect; Pattern : TBitmap));
15543:   Function ValidGraphic( aGraphic : TGraphic) : Boolean);
15544:   Function ColorSetPercentBrightness( Color : TColor; PercentLight : Integer) : TColor);
15545:   Function ColorModify( Color : TColor; incR, incG, incB : Integer) : TColor);
15546:   Function ColorSetPercentContrast( Color : TColor; IncPercent : Integer) : TColor);
15547:   Function ColorSetPercentPale( Color : TColor; IncPercent : integer) : TColor);
15548:   Function MediumColor( Color1, Color2 : TColor) : TColor);
15549:   Function ClientToScreenRect( aControl : TControl; aControlRect : TRect) : TRect);
15550:   Function ScreenToClientRect( aControl : TControl; aScreenRect : TRect) : TRect);
15551:   Function CombineRectKeepingCenterPosition( RectPos, AddRect : TRect) : TRect);
15552:   Procedure InflateRectPercent( var aRect : TRect; withPercent : Double););
15553:   Function GetIntermediateRect( Rect1, Rect2 : TRect; Percent : Double) : TRect);
15554:   Function GetProportionalRect( fromRect, InsideRect : TRect) : TRect);
15555:   Function PointInRect( const aPt : TPoint; const aRect : TRect) : boolean);
15556:   Function PointInEllipse( const aPt : TPoint; const aRect : TRect) : boolean);
15557:   Function CanvasAcceleratorTextWidth( aCanvas : TCanvas; aText : String) : Integer);
15558: end;
15559:
15560: procedure SIRegister_cyTypes(CL: TPSPascalCompiler);
15561: begin
15562:   Type(TGlyphAlignment', '( gaLeft, gaCenter, gaRight )');

```

```

15563:   Type(TGlyphLayout', '( glTop, glCenter, glBottom )');
15564:   Type(TDisabledGlyphOptions', '( dgDoNotDraw, dgDrawNormal, dgDrawMonochrome )');
15565:   Type(TCaptionRender', '( crNormal, crPathEllipsis, crEndEllipsis, crWordEllipsis )');
15566:   Type(TCaptionOrientation', '( coHorizontal, coHorizontalReversed, coVertical, coVerticalReversed )');
15567:   Type(TBgPosition', '( bgCentered, bgTopLeft, bgTopCenter, bgTopRight, '
15568:     + 'bgCenterRight, bgBottomRight, bgBottomCenter, bgBottomLeft, bgCenterLeft )');
15569:   Type(TBgStyle', '( bgNone, bgNormal, bgMosaic, bgStretch, bgStretchProportional )');
15570:   Type(TcyBevelCut', '( bcLowered, bcRaised, bcNone, bcTransparent, bcGradientToNext )');
15571:   Type(TDgradOrientation', '( dgdVertical, dgdHorizontal, dgdAngle, dgdRadial, dgdRectangle )');
15572:   Type(TDgradOrientationShape', '( osRadial, osRectangle )');
15573:   Type(TDgradBalanceMode(bmNormal, bmMirror, bmReverse, bmReverseFromColor, bmInvertReverse,
bmInvertReverseFromColor);
15574:   Type(TRunTimeDesignJob', '( rjNothing, rjMove, rjResizeTop, rjResizeBottom, rjResizeLeft,
rjResizeTopLeft, rjResizeBottomLeft, rjResizeRight, rjResizeTopRight, rjResizeBottomRight )');
15575:   Type(TLineCoord', 'record BottomCoord : TPoint; TopCoord : TPoint; end');
15576:   cCaptionOrientationWarning', 'String')('Note that text orientation doesn't work with all fonts!');
15577: end;
15578:
15579: Functions_max hex in the box maXbox
15580: functionslist.txt
15581: FunctionsList1 3.9.9.86/88/91/92/94
15582:
15583: *****
15584: Procedure
15585: PROCEDURE SIZE 7507 7401 6792 6310 5971 4438 3797 3600
15586: Procedure *****Now the Procedure list*****
15587: Procedure ( ACol, ARow : Integer; Items : TStringList)
15588: Procedure ( Agg : TAggregate)
15589: Procedure ( ASender : TComponent; const AReplyStatus : TReplyStatus)
15590: Procedure ( ASender : TComponent; const AString : String; var AMsg : TIdMessage)
15591: Procedure ( ASender : TComponent; var AMsg : TIdMessage)
15592: Procedure ( ASender : TObject; const ABytes : Integer)
15593: Procedure ( ASender : TObject; VStream : TStream)
15594: Procedure ( AThread : TIdThread)
15595: Procedure ( AWebModule : TComponent)
15596: Procedure ( Column : TColumn)
15597: Procedure ( const AUsername : String; const APassword : String; AAuthenticationResult : Boolean)
15598: Procedure ( const iStart : integer; const sText : string)
15599: Procedure ( Control : TCustomTabControl; TabIndex : Integer; const Rect : TRect; Active : Boolean)
15600: Procedure ( Database : TDatabase; LoginParams : TStringList)
15601: Procedure (DataSet:TCustomClientDataSet;E:EREconcileError;UpdateKind:TUpdateKind;var Action:
TReconcileAction)
15602: Procedure ( DATASET : TDATASET)
15603: Procedure ( DataSet:TDataSet; E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
15604: Procedure ( DATASET : TDATASET; E : TObject; var ACTION : TDATASETACTION)
15605: Procedure ( DataSet : TDataSet; UpdateKind : TUpdateKind; var UpdateAction : TUpdateAction)
15606: Procedure ( DATASET : TDATASET; var ACCEPT : BOOLEAN)
15607: Procedure ( DBCtrlGrid : TDBCtrlGrid; Index : Integer)
15608: Procedure ( Done : Integer)
15609: Procedure ( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
15610: Procedure (HeaderControl:TCustomHeaderControl;Section:THeaderSection; const Rect:TRect;Pressed:Boolean)
15611: Procedure
(HeaderControl:TCustomHeaderControl;Section:THeaderSection;Width:Integer;State:TSectionTrackState)
15612: Procedure ( HeaderControl : THeaderControl; Section : THeaderSection)
15613: Procedure (HeaderControl:THeaderControl;Section:THeaderSection; const Rect:TRect; Pressed : Boolean)
15614: Procedure (HeaderControl:THeaderControl;Section:THeaderSection;Width:Integer;State : TSectionTrackState)
15615: Procedure (Sender: TCustomListView;const ARect: TRect;Stage:TCustomDrawStage;var DefaultDraw: Boolean)
15616: Procedure ( Sender : TCustomListView; const ARect : TRect; var DefaultDraw : Boolean)
15617: Procedure ( Sender : TCustomListView; Item : TListItem; Rect : TRect; State : TOwnerDrawState)
15618: Procedure ( Sender : TCustomTreeView; const ARect : TRect; var DefaultDraw : Boolean)
15619: Procedure ( SENDER : TFIELD; const TEXT : String)
15620: Procedure ( SENDER : TFIELD; var TEXT : STRING; DISPLAYTEXT : BOOLEAN)
15621: Procedure ( Sender : TIdTelnet; const Buffer : String)
15622: Procedure ( Sender : TIdTelnet; Status : TIdTelnetCommand)
15623: Procedure ( SENDER : TOBJECT; ACANVAS : TCanvas; ARECT : TRect; SELECTED : BOOLEAN)
15624: Procedure ( SENDER : TOBJECT; ACANVAS : TCanvas; ARECT : TRect; STATE : TOWNERDRAWSTATE)
15625: Procedure ( SENDER : TOBJECT; ACANVAS : TCanvas; var WIDTH, HEIGHT : INTEGER)
15626: Procedure ( Sender : TObject; ACol, ARow : Longint; const Value : string)
15627: Procedure ( Sender : TObject; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
15628: Procedure ( Sender : TObject; ACol, ARow : Longint; var CanSelect : Boolean)
15629: Procedure ( Sender : TObject; ACol, ARow : Longint; var Value : string)
15630: Procedure ( Sender : TObject; Button : TMPBtnType)
15631: Procedure ( Sender : TObject; Button : TMPBtnType; var DoDefault : Boolean)
15632: Procedure ( Sender : TObject; Button : TUDBtnType)
15633: Procedure ( Sender : TObject; Canvas : TCanvas; PageRect : TRect; var DoneDrawing : Boolean)
15634: Procedure ( Sender: TObject; ClientSocket: TServerClientWinSocket; var SocketThread : TServerClientThread)
15635: Procedure ( Sender : TObject; Column : TListColumn)
15636: Procedure ( Sender : TObject; Column : TListColumn; Point : TPoint)
15637: Procedure ( Sender : TObject; Connecting : Boolean)
15638: Procedure (Sender:TObject;const PapSize:SmallInt;const Orient:TPrtOrient;const PageTy:TPageTy;var
DoneDraw:Bool
15639: Procedure (Sender:TObject; const Rect : TRect; DataCol : Integer; Column : TColumn; State : TGridDrawState)
15640: Procedure ( Sender : TObject; const Rect : TRect; Field : TField; State : TGridDrawState)
15641: Procedure (Sender: TObject; const UserString:string;var DateAndTime:TDateTime;var AllowChange:Boolean)
15642: Procedure ( Sender : TObject; E : Exception; var Handled : Boolean)
15643: Procedure ( Sender : TObject; FromIndex, ToIndex : Longint)
15644: Procedure ( Sender : TObject; FromSection, ToSection : THeaderSection; var AllowDrag : Boolean)
15645: Procedure ( Sender : TObject; Index : Longint)
15646: Procedure ( Sender : TObject; Item : TListItem)

```

```

15647: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange)
15648: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange; var AllowChange : Boolean)
15649: Procedure ( Sender : TObject; Item : TListItem; Selected : Boolean)
15650: Procedure ( Sender : TObject; Item : TListItem; var AllowEdit : Boolean)
15651: Procedure ( Sender : TObject; Item : TListItem; var S : string)
15652: Procedure ( Sender : TObject; Item1, Item2 : TListItem; Data : Integer; var Compare : Integer)
15653: Procedure ( Sender : TObject; ModalResult : TModalResult; var CanClose : Boolean)
15654: Procedure ( Sender : TObject; Month : LongWord; var MonthBoldInfo : LongWord)
15655: Procedure ( Sender : TObject; NewTab : Integer; var AllowChange : Boolean)
15656: Procedure ( Sender : TObject; Node : TTreeNode)
15657: Procedure ( Sender : TObject; Node : TTreeNode; var AllowChange : Boolean)
15658: Procedure ( Sender : TObject; Node : TTreeNode; var AllowCollapse : Boolean)
15659: Procedure ( Sender : TObject; Node : TTreeNode; var AllowEdit : Boolean)
15660: Procedure ( Sender : TObject; Node : TTreeNode; var AllowExpansion : Boolean)
15661: Procedure ( Sender : TObject; Node : TTreeNode; var S : string)
15662: Procedure ( Sender : TObject; Node1, Node2 : TTreeNode; Data : Integer; var Compare : Integer)
15663: Procedure ( Sender : TObject; NumObjects, NumChars : Integer; var SaveClipboard : Boolean)
15664: Procedure ( Sender : TObject; Rect : TRect)
15665: Procedure ( Sender : TObject; Request : TWebRequest; Response : TWebResponse; var Handled : Boolean)
15666: Procedure (Sender:TObject;Shift:TShiftState;X,Y:Integer;Orient:TPageScrollerOrientation;var Delta:Int
15667: Procedure ( Sender : TObject; Socket : TCustomWinSocket)
15668: Procedure (Sender:TObject; Socket:TCustomWinSocket;ErrorEvent: TErrorEvent; var ErrorCode : Integer)
15669: Procedure ( Sender : TObject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
15670: Procedure ( Sender : TObject; Socket : TSocket; var ClientSocket : TServerClientWinSocket)
15671: Procedure ( SENDER : TOBJECT; SOURCE : TMENUITEM; REBUILD : BOOLEAN)
15672: Procedure ( Sender : TObject; StartPos, EndPos : Integer; var AllowChange : Boolean)
15673: Procedure ( Sender : TObject; TabCanvas : TCanvas; R : TRect; Index : Integer; Selected : Boolean)
15674: Procedure ( Sender : TObject; TabIndex : Integer; var ImageIndex : Integer)
15675: Procedure ( Sender : TObject; Thread : TServerClientThread)
15676: Procedure ( Sender : TObject; TickCount : Cardinal; var Reset : Boolean)
15677: Procedure ( Sender : TObject; Username, Password : string)
15678: Procedure ( Sender : TObject; var AllowChange : Boolean)
15679: Procedure ( Sender : TObject; var AllowChange : Boolean; NewValue : SmallInt; Direction:TUpDownDirection)
15680: Procedure ( Sender : TObject; var Caption : string; var Alignment : THTMLCaptionAlignment)
15681: Procedure ( Sender : TObject; var Continue : Boolean)
15682: Procedure (Sender:TObject;var dest:string;var NumRedirect:Int;var Handled:bool; var VMethod:TidHTTPMethod)
15683: Procedure ( Sender : TObject; var Username : string)
15684: Procedure ( Sender : TObject; Wnd : HWND)
15685: Procedure ( Sender : TToolBar; Button : TToolButton)
15686: Procedure ( Sender : TToolBar; const ARect : TRect; Stage : TCustomDrawStage; var DefaultDraw : Boolean)
15687: Procedure ( Sender : TToolBar; const ARect : TRect; var DefaultDraw : Boolean)
15688: Procedure ( Sender : TToolBar; Index : Integer; var Allow : Boolean)
15689: Procedure ( Sender : TToolBar; Index : Integer; var Button : TToolButton)
15690: Procedure ( StatusBar : TCustomStatusBar; Panel : TStatusPanel; const Rect : TRect)
15691: Procedure ( StatusBar : TStatusBar; Panel : TStatusPanel; const Rect : TRect)
15692: Procedure (var FieldNames:TWideStrings;SQL:WideString; var BindAllFields:Boolean;var TableName: WideString)
15693: Procedure ( var FieldNames : TWideStrings; SQL : WideString; var TableName : WideString)
15694: procedure (Sender: TObject)
15695: procedure (Sender: TObject; var Done: Boolean)
15696: procedure (Sender: TObject; var Key: Word; Shift: TShiftState);
15697: procedure _T(Name: tbtString; v: Variant);
15698: Procedure AbandonSignalHandler( RtlSigNum : Integer)
15699: Procedure Abort
15700: Procedure About1Click( Sender : TObject)
15701: Procedure Accept( Socket : TSocket)
15702: Procedure AESSymmetricExecute(const plaintext, ciphertext, password: string)
15703: Procedure AESEncryptFile(const plaintext, ciphertext, password: string)
15704: Procedure AESDecryptFile(const replaintext, ciphertext, password: string)
15705: Procedure AESEncryptString(const plaintext: string; var ciphertext: string; password: string)
15706: Procedure AESDecryptString(var plaintext: string; const ciphertext: string; password: string)
15707: Procedure Add( Addend1, Addend2 : TMyBigInt)
15708: Procedure ADD( const AKEY, AVALUE : VARIANT)
15709: Procedure Add( const Key : string; Value : Integer)
15710: Procedure ADD( const NAME, FIELDS : string; OPTIONS : TINDEXTIONS)
15711: Procedure ADD( FIELD : TFIELD)
15712: Procedure ADD( ITEM : TMENUITEM)
15713: Procedure ADD( POPUP : TPOPMENU)
15714: Procedure AddCharacters( xCharacters : TCharSet)
15715: Procedure AddDriver( const Name : string; List : TStringList)
15716: Procedure AddImages( Value : TCustomImageList)
15717: Procedure AddIndex( const Name, Fields : string; Options : TIndexOptions; const DescFields : string)
15718: Procedure AddLambdaTransitionTo( oState : TniRegularExpressionState)
15719: Procedure AddLoader( Loader : TBitmapLoader)
15720: Procedure ADDPARAM( VALUE : TPARAM)
15721: Procedure AddPassword( const Password : string)
15722: Procedure AddStandardAlias( const Name, Path, DefaultDriver : string)
15723: Procedure AddState( oState : TniRegularExpressionState)
15724: Procedure AddStrings( Strings : TStringList);
15725: procedure AddStrings(Strings: TStringList);
15726: Procedure AddStrings1( Strings : TWideStrings);
15727: Procedure AddStringTerm( var sString : string; const sTerm : string; const sSeparator : string)
15728: Procedure AddToRecentDocs( const Filename : string)
15729: Procedure AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharSet)
15730: Procedure AllFunctionsList1Click( Sender : TObject)
15731: procedure AllObjectsList1Click(Sender: TObject);
15732: Procedure Allocate( AAllocateBytes : Integer)
15733: procedure AllResourceList1Click(Sender: TObject);
15734: Procedure AnsiAppend( var dst : AnsiString; const src : AnsiString)
15735: Procedure AnsiAssign( var dst : AnsiString; var src : AnsiString)

```

```

15736: Procedure AnsiDelete( var dst : AnsiString; index, count : Integer)
15737: Procedure AnsiFree( var s : AnsiString)
15738: Procedure AnsiFromWide( var dst : AnsiString; const src : WideString)
15739: Procedure AnsiInsert( var dst : AnsiString; const src : AnsiString; index : Integer)
15740: Procedure AnsiSetLength( var dst : AnsiString; len : Integer)
15741: Procedure AnsiString_to_stream( const Value : ansistring; Destin : TStream)
15742: Procedure AntiFreeze;
15743: Procedure APPEND
15744: Procedure Append( const S : WideString)
15745: procedure Append(S: string);
15746: Procedure AppendByte( var VBytes : TIdBytes; AByte : byte)
15747: Procedure AppendBytes( var VBytes : TIdBytes; AAdd : TIdBytes)
15748: Procedure AppendChunk( Val : OleVariant)
15749: Procedure AppendData( const Data : OleVariant; HitEOF : Boolean)
15750: Procedure AppendStr( var Dest : string; S : string)
15751: Procedure AppendString( var VBytes : TIdBytes; const AStr : String; ALength : Integer)
15752: Procedure ApplyRange
15753: procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
15754: Procedure Arrange( Code : TListArrangement)
15755: procedure Assert(expr : Boolean; const msg: string);
15756: procedure Assert2(expr : Boolean; const msg: string);
15757: Procedure Assign( AList : TCustomBucketList)
15758: Procedure Assign( Other : TObject)
15759: Procedure Assign( Source : TDragObject)
15760: Procedure Assign( Source : TPersistent)
15761: Procedure Assign(Source: TPersistent)
15762: procedure Assign2(mystring, mypath: string);
15763: Procedure AssignCurValues( Source : TDataSet);
15764: Procedure AssignCurValues1( const CurValues : Variant);
15765: Procedure ASSIGNFIELD( FIELD : TFIELD)
15766: Procedure ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT)
15767: Procedure AssignFile(var F: Text; FileName: string)
15768: procedure AssignFile(var F: TextFile; FileName: string)
15769: procedure AssignFileRead(var mystring, myfilename: string);
15770: procedure AssignFileWrite(mystring, myfilename: string);
15771: Procedure AssignTo( Other : TObject)
15772: Procedure AssignValues( Value : TParameters)
15773: Procedure ASSIGNVALUES( VALUE : TPARAMS)
15774: Procedure AssociateExtension( IconPath, ProgramName, Path, Extension : string)
15775: Procedure Base64_to_stream( const Base64 : ansistring; Destin : TStream)
15776: Procedure Base64ToVar( NatData : Pointer; const SoapData : WideString);
15777: Procedure Base64ToVar1( var V : Variant; const SoapData : WideString);
15778: Procedure BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
15779: Procedure BcdDivide( Dividend, Divisor : string; var bcdOut : TBcd);
15780: Procedure BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
15781: Procedure BcdDivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
15782: Procedure BcdDivide3( const Dividend : TBcd; const Divisor : string; var bcdOut : TBcd);
15783: Procedure BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
15784: Procedure BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
15785: Procedure BcdMultiply2( const bcdIn : TBcd; const StringIn : string; var bcdOut : TBcd);
15786: Procedure BcdMultiply3( StringIn1, StringIn2 : string; var bcdOut : TBcd);
15787: Procedure BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
15788: Procedure BcdToBytes( Value : TBcd; Bytes : array of byte)
15789: procedure Beep
15790: Procedure BeepOk
15791: Procedure BeepQuestion
15792: Procedure BeepHand
15793: Procedure BeepExclamation
15794: Procedure BeepAsterisk
15795: Procedure BeepInformation
15796: procedure BEGINDRAG(IMMEDIATE:BOOLEAN)
15797: Procedure BeginLayout
15798: Procedure BeginTimer( const Delay, Resolution : Cardinal)
15799: Procedure BeginUpdate
15800: procedure BeginUpdate;
15801: procedure BigScreen1Click(Sender: TObject);
15802: procedure BinToHex(Buffer: PChar; Text: PChar; BufSize: Integer);
15803: Procedure BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits : Boolean);
15804: Procedure BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits : Boolean);
15805: Procedure BitsToBooleans2( const Bits : Integer; var B : TBooleanArray; AllBits : Boolean);
15806: Procedure BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits : Boolean);
15807: Procedure BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord)
15808: Procedure BooleansToBits( var Dest : Byte; const B : TBooleanArray);
15809: Procedure BooleansToBits1( var Dest : Word; const B : TBooleanArray);
15810: Procedure BooleansToBits2( var Dest : Integer; const B : TBooleanArray);
15811: Procedure BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
15812: Procedure BreakPointMenuClick( Sender : TObject)
15813: procedure BRINGTOFRONT
15814: procedure BringToFront;
15815: Procedure btnBackClick( Sender : TObject)
15816: Procedure btnBrowseClick( Sender : TObject)
15817: Procedure BtnClick( Index : TNavigateBtn)
15818: Procedure btnLargeIconsClick( Sender : TObject)
15819: Procedure BuildAndSendRequest( AURI : TIdURI)
15820: Procedure BuildCache
15821: Procedure BurnMemory( var Buff, BuffLen : integer)
15822: Procedure BurnMemoryStream( Destructo : TMemoryStream)
15823: Procedure CalculateFirstSet
15824: Procedure Cancel

```



```

15825: procedure CancelDrag;
15826: Procedure CancelEdit
15827: procedure CANCELHINT
15828: Procedure CancelRange
15829: Procedure CancelUpdates
15830: Procedure CancelWriteBuffer
15831: Procedure Capture1(ADest:TStream;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool;
15832: Capture2( ADest : TStrings; const ADelim : string; const AIsRFCMessage : Boolean);
15833: Procedure Capture3(ADest:TStrings;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool
15834: procedure CaptureScreenFormat(vname: string; vextension: string);
15835: procedure CaptureScreenPNG(vname: string);
15836: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
15837: procedure CASCADE
15838: Procedure CastNativeToSoap(Info:PTypeInfo;var SoapData:WideString;NatData:Pointer;var IsNull: Boolean)
15839: Procedure CastSoapToVariant( SoapInfo : PTypeInfo; const SoapData : WideString; NatData : Pointer);
15840: Procedure cbPathClick( Sender : TObject)
15841: Procedure cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
15842: Procedure cedebugAfterExecute( Sender : TPSScript)
15843: Procedure cedebugBreakpoint( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
15844: Procedure cedebugCompile( Sender : TPSScript)
15845: Procedure cedebugExecute( Sender : TPSScript)
15846: Procedure cedebugIdle( Sender : TObject)
15847: Procedure cedebugLineInfo( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
15848: Procedure CenterHeight( const pc, pcParent : TControl)
15849: Procedure CenterDlg(AForm: TForm; MForm: TForm); { Zentriert Forms }
15850: Procedure CenterForm(AForm: TForm; MForm: TForm); { Zentriert Forms }
15851: Procedure Change
15852: procedure ChangeBiDiModeAlignment(var Alignment: TAlignment);
15853: Procedure Changed
15854: Procedure ChangeDir( const ADirName : string)
15855: Procedure ChangeDirUp
15856: Procedure ChangeEntryTransitions( oNewState : TniRegularExpressionState)
15857: Procedure ChangeLevelBy( Value : TChangeRange)
15858: Procedure ChDir(const s: string)
15859: Procedure Check(Status: Integer)
15860: Procedure CheckCommonControl( CC : Integer)
15861: Procedure CHECKFIELDNAME( const FIELDNAME : String)
15862: Procedure CHECKFIELDNAMES( const FIELDNAMES : String)
15863: Procedure CheckForDisconnect(const ARaiseExceptionIfDisconnected: boolean;const AIgnoreBuffer:bool)
15864: Procedure CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected : Boolean)
15865: Procedure CheckToken( T : Char)
15866: procedure CheckToken(t:char)
15867: Procedure CheckTokenSymbol( const S : string)
15868: procedure CheckTokenSymbol(s:string)
15869: Procedure CheckToolMenuDropdown( ToolButton : TToolButton)
15870: procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
15871: Procedure CIED65ToCIED50( var X, Y, Z : Extended)
15872: Procedure CIELABToBGR( const Source, Target : Pointer; const Count : Cardinal);
15873: procedure CipherFile1Click(Sender: TObject);
15874: Procedure Clear;
15875: Procedure Clear1Click( Sender : TObject)
15876: Procedure ClearColor( Color : TColor)
15877: Procedure CLEARITEM( AITEM : TMENUITEM)
15878: Procedure ClearMapping
15879: Procedure ClearSelection( KeepPrimary : Boolean)
15880: Procedure ClearWriteBuffer
15881: Procedure Click
15882: Procedure Close
15883: Procedure Close1Click( Sender : TObject)
15884: Procedure CloseDatabase( Database : TDatabase)
15885: Procedure CloseDataSets
15886: Procedure CloseDialog
15887: Procedure CloseFile(var F: Text);
15888: Procedure Closure
15889: Procedure CMYKToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
15890: Procedure CMYKToBGR1( const C, M, Y, K, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
15891: Procedure CodeCompletionList1Click( Sender : TObject)
15892: Procedure ColEnter
15893: Procedure Collapse
15894: Procedure Collapse( Recurse : Boolean)
15895: Procedure ColorRGBToHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word)
15896: Procedure CommaSeparatedToStringList( AList : TStrings; const Value : string)
15897: Procedure CommitFreeAndNil( var Transaction : TDBXTransaction)
15898: Procedure Compile1Click( Sender : TObject)
15899: procedure ComponentCount1Click(Sender: TObject);
15900: Procedure Compress(azipfolder, azipfile: string)
15901: Procedure DeCompress(azipfolder, azipfile: string)
15902: Procedure XZip(azipfolder, azipfile: string)
15903: Procedure XUnZip(azipfolder, azipfile: string)
15904: Procedure Connect( const ATimeout : Integer)
15905: Procedure Connect( Socket : TSocket)
15906: procedure Console1Click(Sender: TObject);
15907: Procedure Continue
15908: Procedure ContinueCount( var Counter : TJclCounter)
15909: procedure CONTROLDESTROYED(CONTROL:TCONTROL)
15910: Procedure ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : integer)
15911: Procedure ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : integer)
15912: Procedure ConvertImage(vsource, vdestination: string);
15913: // Ex. ConvertImage(Exepath+'my233_bmp.bmp',Exepath+'mypng111.png')

```

```

15914: Procedure ConvertToGray(Cnv: TCanvas);
15915: Procedure Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Integer; Length : Integer)
15916: Procedure Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Integer; Count : Integer);
15917: Procedure Copy1( Source : TBytes; Offset : Integer; Buffer : TValueBuffer; Count : Integer);
15918: Procedure CopyBytesToHostLongWord(const ASource:TIdBytes;const ASourceIndex:Integer;var VDest:LongWord)
15919: Procedure CopyBytesToHostWord( const ASource : TIdBytes; const ASourceIndex : Integer; var VDest : Word)
15920: Procedure CopyFrom( mbCopy : TMyBigInt)
15921: Procedure CopyMemoryStream( Source, Destination : TMemoryStream)
15922: procedure CopyRect(const Dest: TRect; Canvas: TCanvas;const Source: TRect);
15923: Procedure CopyTidByteArray( const ASource : array of Byte; const ASourceIndex : Integer; var VDest : array
of Byte; const ADestIndex : Integer; const ALength : Integer)
15924: Procedure CopyTidBytes(const ASrc:TIdBytes;const ASrcIdx:Int;var VDest:TIdBytes;const ADestIdx:Int;const
ALen:Int)
15925: Procedure CopyTidCardinal( const ASource : Cardinal; var VDest : TIdBytes; const ADestIndex : Integer)
15926: Procedure CopyTidInt64( const ASource : Int64; var VDest : TIdBytes; const ADestIndex : Integer)
15927: Procedure CopyTidIPv6Address(const ASource:TIdIPv6Address; var VDest: TIdBytes; const ADestIndex : Integer)
15928: Procedure CopyTidLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex : Integer)
15929: Procedure CopyTidNetworkLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex:Integer)
15930: Procedure CopyTidNetworkWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
15931: Procedure CopyTidString(const ASource:String;var VDest:TIdBytes;const ADestIndex:Integer;ALength: Integer)
15932: Procedure CopyTidWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
15933: Procedure CopyToClipboard
15934: Procedure CountParts
15935: Procedure CreateDataSet
15936: Procedure CreateEmptyFile( const FileName : string)
15937: Procedure CreateFileFromString( const FileName, Data : string)
15938: Procedure CreateFromDelta( Source : TPacketDataSet)
15939: procedure CREATEHANDLE
15940: Procedure CreateProcAsUser( const UserDomain, UserName, Password, CommandLine : string)
15941: Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const Environment:PChar)
15942: Procedure CreateTable
15943: Procedure CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString)
15944: procedure CSyntax1Click(Sender: TObject);
15945: Procedure CurrencyToComp( Value : Currency; var Result : Comp)
15946: Procedure CURSORPOSCHANGED
15947: procedure CutFirstDirectory(var S: String)
15948: Procedure DataBaseError(const Message: string)
15949: Procedure DateTimeToString( var Result : string; Format : string; DateTime : TDateTime);
15950: procedure DateTimeToString(var Result: string; const Format: string; DateTime: TDateTime)
15951: Procedure DateTimeToSystemTime( DateTime : TDateTime; var SystemTime : TSystemTime)
15952: procedure DateTimeToSystemTime(const DateTime: TDateTime; var SystemTime: TSystemTime);
15953: Procedure DBIError(errorCode: Integer)
15954: Procedure DebugOutput( const AText : string)
15955: Procedure DebugRun1Click( Sender : TObject)
15956: procedure Dec;
15957: Procedure DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word)
15958: procedure DecodeDate(const DateTime: TDateTime; var Year, Month, Day: Word);
15959: Procedure DecodeDateDay( const AValue : TDateTime; out AYear, ADayOfYear : Word)
15960: Procedure DecodeDateMonthWeek(const AValue:TDateTime;out AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
15961: Procedure DecodeDateTime(const AValue:TDateTime;out AYear, AMonth, ADay, AHour, AMin, ASec, AMillSec:Word)
15962: Procedure DecodeDateWeek( const AValue : TDateTime; out AYear, AWeekOfYear, ADayOfWeek : Word)
15963: Procedure DecodeDayOfWeekInMonth(const AValue:TDateTime;out AYear, AMonth, ANthDayOfWeek, ADayOfWeek:Word)
15964: Procedure DecodeTime( DateTime : TDateTime; var Hour, Min, Sec, MSec : Word)
15965: procedure DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word);
15966: Procedure Decompile1Click( Sender : TObject)
15967: Procedure DefaultDrawColumnCell(const Rect : TRect; DataCol:Integer;Column:TColumn;State:TGridDrawState)
15968: Procedure DefaultDrawDataCell( const Rect : TRect; Field : TField; State : TGridDrawState)
15969: Procedure DeferLayout
15970: Procedure deffileread
15971: procedure DEFOCUSCONTROL(CONTROL:TWINCONTROL;REMOVING:BOOLEAN)
15972: Procedure DelayMicroseconds( const MicroSeconds : Integer)
15973: Procedure Delete
15974: Procedure Delete( const AFilename : string)
15975: Procedure Delete( const Index : Integer)
15976: Procedure DELETE( INDEX : INTEGER)
15977: Procedure Delete( Index : LongInt)
15978: Procedure Delete( Node : TTreeNode)
15979: procedure Delete(var s: AnyString; ifrom, icount: Longint);
15980: Procedure DeleteAlias( const Name : string)
15981: Procedure DeleteDriver( const Name : string)
15982: Procedure DeleteIndex( const Name : string)
15983: Procedure DeleteKey( const Section, Ident : String)
15984: Procedure DeleteRecords
15985: Procedure DeleteRecords( AffectRecords : TAffectRecords)
15986: Procedure DeleteString( var pStr : String; const pDelStr : string)
15987: Procedure DeleteTable
15988: procedure DelphiSite1Click(Sender: TObject);
15989: Procedure Deselect
15990: Procedure Deselect( Node : TTreeNode)
15991: procedure DestroyComponents
15992: Procedure DestroyHandle
15993: Procedure Diff( var X : array of Double)
15994: procedure Diff(var X: array of Double);
15995: procedure DISABLEALIGN
15996: Procedure DisableConstraints
15997: Procedure Disconnect
15998: Procedure Disconnect( Socket : TSocket)
15999: Procedure Dispose
16000: procedure Dispose(P: PChar)

```

```

16001: Procedure DivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
16002: Procedure DoKey( Key : TDBCtrlGridKey)
16003: Procedure DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
16004: Procedure DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
16005: Procedure Dormant
16006: Procedure DoubleToBcd( const AValue : Double; var bcd : TBcd);
16007: Procedure DoubleToBytes( Value : Double; Bytes : array of byte)
16008: Procedure DoubleToComp( Value : Double; var Result : Comp)
16009: Procedure Draw( Canvas : TCanvas; X, Y, Index : Integer; Enabled : Boolean);
16010: procedure Draw(X, Y: Integer; Graphic: TGraphic);
16011: Procedure Draw1(Canvas:TCanvas; X,Y,
Index:Int;ADrawingStyle:TDrawingStyle;AImageType:TImageType;Enabled:Bool);
16012: Procedure DrawArrow( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
16013: Procedure DrawCheck( ACanvas : TCanvas; Location : TPoint; Size : Integer; Shadow : Boolean)
16014: Procedure DrawChevron( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
16015: Procedure DrawColumnCell( const Rect : TRect; DataCol : Integer; Column : TColumn; State : TGridDrawState)
16016: procedure DrawFocusRect(const Rect: TRect);
16017: Procedure DrawHBIToTBitmap( HDIB : THandle; Bitmap : TBitmap)
16018: Procedure DRAWMENUITEM( MENUITEM : TMENUITEM; ACANVAS : TCanvas; ARECT : TRect; STATE : TOWNERDRAWSTATE)
16019: Procedure DrawOverlay(Canvas:TCanvas;X,Y:Integer;ImageIndex:Integer;Overlay:TOverlay;Enabled: Boolean);
16020: Procedure DrawOverlay1(Canvas:TCanvas; X,Y:Int; ImageIndex:Int; Overlay : TOverlay; ADrawingStyle :
TDrawingStyle; AImageType : TImageType; Enabled : Boolean);
16021: procedure drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: integer);
16022: Procedure DrawPolyLine( const Canvas : TCanvas; var Points : TPointArray; const ClipRect : TRect)
16023: Procedure DropConnections
16024: Procedure DropDown
16025: Procedure DumpDescription( oStrings : TStrings)
16026: Procedure DumpStateTable( oStrings : TStrings)
16027: Procedure EDIT
16028: Procedure EditButtonClick
16029: Procedure EditFont1Click( Sender : TObject)
16030: procedure Ellipse(X1, Y1, X2, Y2: Integer);
16031: Procedure Ellipse1( const Rect : TRect);
16032: Procedure EMMS
16033: Procedure Encode( ADest : TStream)
16034: procedure ENDDRAG(DROP:BOOLEAN)
16035: Procedure EndEdit( Cancel : Boolean)
16036: Procedure EndTimer
16037: Procedure EndUpdate
16038: Procedure EraseSection( const Section : string)
16039: Procedure Error( const Ident : string)
16040: procedure Error(Ident:Integer)
16041: Procedure ErrorFmt( const Ident : string; const Args : array of const)
16042: Procedure ErrorStr( const Message : string)
16043: procedure ErrorStr(Message:String)
16044: Procedure Exchange( Index1, Index2 : Integer)
16045: procedure Exchange(Index1, Index2: Integer);
16046: Procedure Exec( FileName, Parameters, Directory : string)
16047: Procedure ExecProc
16048: Procedure ExecSQL( UpdateKind : TUpdateKind)
16049: Procedure Execute
16050: Procedure Execute( const CommandText : WideString; var Params, OwnerData : OleVariant)
16051: Procedure ExecuteAndWait( FileName : string; Visibility : Integer)
16052: Procedure ExecuteCommand(executeFile, paramstring: string)
16053: Procedure ExecuteShell(executeFile, paramstring: string)
16054: Procedure ExitThread(ExitCode: Integer); stdcall;
16055: Procedure ExitProcess(ExitCode: Integer); stdcall;
16056: Procedure Expand( AUserName : string; AResults : TStrings)
16057: Procedure Expand( Recurse : Boolean)
16058: Procedure ExportClipboard1Click( Sender : TObject)
16059: Procedure ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress)
16060: Procedure ExtractContentFields( Strings : TStrings)
16061: Procedure ExtractCookieFields( Strings : TStrings)
16062: Procedure ExtractFields( Separators, WhiteSpace : TSysCharSet; Content : PChar; Strings : TStrings)
16063: Procedure ExtractHeaderFields(Separ,
WhiteSpace:TSysChSet;Cont:PChar;Strings:TStrings;Decode:Bool;StripQuots:Bool)
16064: Procedure ExtractHTTFFields(Separators,WhiteSpace:
TSysCharSet;Content:PChar;Strings:TStrings;StripQuots:Bool)
16065: Procedure ExtractQueryFields( Strings : TStrings)
16066: Procedure FastDegToGrad
16067: Procedure FastDegToRad
16068: Procedure FastGradToDeg
16069: Procedure FastGradToRad
16070: Procedure FastRadToDeg
16071: Procedure FastRadToGrad
16072: Procedure FileClose( Handle : Integer)
16073: Procedure FileClose(handle: integer)
16074: procedure FilesFromWildcard(Dir,Mask:string;var Files:TStringList;Subdirs,ShowDirs,Multitasking:Bool)
16075: Procedure FileStructure( AStructure : TIdFTPDataStructure)
16076: Procedure FillByte2(var X: Byte ; count: integer; value: byte)
16077: Procedure FillBytes( var VBytes : TIdBytes; const ACount : Integer; const AValue : Byte)
16078: Procedure FillChar( Buffer : TRecordBuffer; Length : Integer; value : Byte)
16079: Procedure FillChar2(var X: PChar ; count: integer; value: char)
16080: Procedure FillCharS(var p: string; count: integer; value: char); //fix3.8
16081: Procedure FillIPLList
16082: procedure FillRect(const Rect: TRect);
16083: Procedure FillTStrings( AStrings : TStrings)
16084: Procedure FilterOnBookmarks( Bookmarks : array of const)
16085: procedure FinalizePackage(Module: HMODULE)

```

```

16086: procedure FindClose;
16087: procedure FindClose2(var F: TSearchRec)
16088: Procedure FindMatches( const sString : string; xNotify : TniRegularExpressionMatchFoundEvent);
16089: Procedure FindMatches1(const sString:string;iStart:integer;xNotify:TniRegularExpressionMatchFoundEvent);
16090: Procedure FindNearest( const KeyValues : array of const)
16091: Procedure FinishContext
16092: Procedure FIRST
16093: Procedure FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float)
16094: Procedure FloatToDecimal(var Result:TFloatRec;const Value:extend;ValueType:TFloatValue;Precis,Decs:Int);
16095: Procedure FloodFill( X, Y : Integer; Color : TColor; FillStyle : TFillStyle)
16096: Procedure FlushSchemaCache( const TableName : string)
16097: procedure FmtStr(var Result: string; const Format: string; const Args: array of const)
16098: procedure FOCUSCONTROL(CONTROL:TWINCONTROL)
16099: Procedure FormlClose( Sender : TObject; var Action : TCloseAction)
16100: Procedure FormActivate( Sender : TObject)
16101: procedure FormatLn(const format: string; const args: array of const); //alias
16102: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
16103: Procedure FormCreate( Sender : TObject)
16104: Procedure FormDestroy( Sender : TObject)
16105: Procedure FormKeyPress( Sender : TObject; var Key : Char)
16106: procedure FormOutput1Click(Sender: TObject);
16107: Procedure FormToHtml( Form : TForm; Path : string)
16108: procedure FrameRect(const Rect: TRect);
16109: Procedure Frame3D(Canvas:TCanvas; var Rect:TRect; TopColor,BottomColor:TColor; Width : Integer)
16110: Procedure NotebookHandlesNeeded( Notebook : TNotebook)
16111: Procedure Free( Buffer : TRecordBuffer)
16112: Procedure Free( Buffer : TValueBuffer)
16113: Procedure Free;
16114: Procedure FreeAndNil(var Obj:TObject)
16115: Procedure FreeImage
16116: procedure FreeMem(P: PChar; Size: Integer)
16117: Procedure FreeTreeData( Tree : TUpdateTree)
16118: Procedure Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Integer)
16119: Procedure FullCollapse
16120: Procedure FullExpand
16121: Procedure GenerateDPB(sl: TStrings; var DPB: string; var DPBLength: Short); //InterBase
16122: Procedure GenerateTPB(sl: TStrings; var TPB: string; var TPBLength: Short);
16123: Procedure OUTPUTXML( SQLOBJECT : TIBSQL; OUTPUTOBJECT : TIBOUTPUTXML)
16124: Procedure Get1( AURL : string; const AResponseContent : TStream);
16125: Procedure Get1( const ASourceFile : string; ADest : TStream; AResume : Boolean);
16126: Procedure Get2(const ASourceFile,ADestFile: string;const ACanOverwrite: boolean; AResume: Boolean);
16127: Procedure GetAliasNames( List : TStrings)
16128: Procedure GetAliasParams( const AliasName : string; List : TStrings)
16129: Procedure GetApplicationsRunning( Strings : TStrings)
16130: Procedure GetCommandTypes( List : TWideStrings)
16131: Procedure GetConfigParams( const Path, Section : string; List : TStrings)
16132: Procedure GetConnectionNames( List : TStrings; Driver : string; DesignMode : Boolean)
16133: Procedure GetConvFamilies( out AFamilies : TConvFamilyArray)
16134: Procedure GetConvTypes( const AFamily : TConvFamily; out ATypes : TConvTypeArray)
16135: Procedure GetDatabaseNames( List : TStrings)
16136: Procedure GetDataPacket( DataSet : TDataSet; var RecsOut : Integer; out Data : OleVariant)
16137: Procedure GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize: longWORD; var ImageSize : longWORD)
16138: Procedure GetDir(d: byte; var s: string)
16139: Procedure GetDirList( const Search : string; List : TStrings; Recursive : Boolean)
16140: Procedure GetDriverNames( List : TStrings)
16141: Procedure GetDriverNames( List : TStrings; DesignMode : Boolean)
16142: Procedure GetDriverParams( const DriverName : string; List : TStrings)
16143: Procedure GetEmails1Click( Sender : TObject)
16144: Procedure getEnvironmentInfo;
16145: Function getEnvironmentString: string;
16146: Procedure GetFieldNames( const DatabaseName, TableName : string; List : TStrings)
16147: Procedure GetFieldNames( const TableName : string; List : TStrings)
16148: Procedure GetFieldNames( const TableName : string; List : TStrings);
16149: Procedure GetFieldNames( const TableName : WideString; List : TWideStrings);
16150: Procedure GETFIELDNAMES( LIST : TSTRINGS)
16151: Procedure GetFieldNames1( const TableName : string; List : TStrings);
16152: Procedure GetFieldNames1( const TableName : string; SchemaName : string; List : TStrings);
16153: Procedure GetFieldNames2( const TableName : WideString; SchemaName : WideString; List : TWideStrings);
16154: Procedure GetFieldNames3( const TableName : WideString; List : TWideStrings);
16155: Procedure GetFileAttributeList( const Items : TStrings; const Attr : Integer)
16156: Procedure GetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
16157: Procedure GetFMTBcd( Buffer : TRecordBuffer; var value : TBcd)
16158: Procedure GetFormatSettings
16159: Procedure GetFromDIB( var DIB : TBitmapInfo)
16160: Procedure GetFromHDIB( HDIB : HBitmap)
16161: Procedure GetIcon( Index : Integer; Image : TIcon);
16162: Procedure GetIcon1(Index : Integer; Image:TIcon; ADrawingStyle:TDrawingStyle; AImageType:TImageType);
16163: Procedure GetIndexInfo( IndexName : string)
16164: Procedure GetIndexNames( const TableName, SchemaName : string; List : TStrings);
16165: Procedure GetIndexNames( List : TStrings)
16166: Procedure GetIndexNames1( const TableName : WideString; List : TWideStrings);
16167: Procedure GetIndexNames2( const TableName, SchemaName : WideString; List : TWideStrings);
16168: Procedure GetIndexNames4( const TableName : string; List : TStrings);
16169: Procedure GetInternalResponse
16170: Procedure GETITEMNAMES( LIST : TSTRINGS)
16171: procedure GetMem(P: PChar; Size: Integer)
16172: Procedure GETOLE2ACCELERATORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:INTEGER;GROUPS:array of INTEGER)
16173: procedure GetPackageDescription(ModuleName: PChar): string)
16174: Procedure GetPackageNames( List : TStrings);

```



```

16175: Procedure GetPackageNames1( List : TWideStrings);
16176: Procedure GetParamList( List : TList; const ParamNames : WideString)
16177: Procedure GetProcedureNames( List : TStringList);
16178: Procedure GetProcedureNames( List : TWideStrings);
16179: Procedure GetProcedureNames1( const PackageName : string; List : TStringList);
16180: Procedure GetProcedureNames1( List : TStringList);
16181: Procedure GetProcedureNames2( const PackageName, SchemaName : string; List : TStringList);
16182: Procedure GetProcedureNames3( List : TWideStrings);
16183: Procedure GetProcedureNames4( const PackageName : WideString; List : TWideStrings);
16184: Procedure GetProcedureNames5( const PackageName, SchemaName : WideString; List : TWideStrings);
16185: Procedure GetProcedureParams( ProcedureName : WideString; List : TList);
16186: Procedure GetProcedureParams1( ProcedureName, PackageName : WideString; List : TList);
16187: Procedure GetProcedureParams2( ProcedureName, PackageName, SchemaName : WideString; List : TList);
16188: Procedure GetProviderNames( Names : TWideStrings);
16189: Procedure GetProviderNames( Proc : TGetStrProc);
16190: Procedure GetProviderNames1( Names : TStringList);
16191: procedure GetQRCode2(Width,Height:Word; Correct_Level: string; const Data:string; apath: string);
16192: procedure GetQRCode3(Width,Height:Word;Correct_Level:string;const Data:string;apath:string);//no autoopen
image
16193: Function GetQRCode4(Width,Height:Word;Correct_Level:string; const
Data:string;aformat:string):TLinearBitmap;
16194: Procedure GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte)
16195: Procedure GetSchemaNames( List : TStringList);
16196: Procedure GetSchemaNames1( List : TWideStrings);
16197: Procedure GetSessionNames( List : TStringList);
16198: Procedure GetStoredProcNames( const DatabaseName : string; List : TStringList);
16199: Procedure GetStrings( List : TStringList);
16200: Procedure GetSystemTime; stdcall;
16201: Procedure GetTableNames(const DatabaseName,Pattern:string;Extensions,SystemTables:Boolean;List:TStringList)
16202: Procedure GetTableNames( List : TStringList; SystemTables : Boolean)
16203: Procedure GetTableNames( List : TStringList; SystemTables : Boolean);
16204: Procedure GetTableNames( List : TWideStrings; SystemTables : Boolean);
16205: Procedure GetTableNames1( List : TStringList; SchemaName : WideString; SystemTables : Boolean);
16206: Procedure GetTableNames1( List : TStringList; SystemTables : Boolean);
16207: Procedure GetTableNames2( List : TWideStrings; SchemaName : WideString; SystemTables : Boolean);
16208: Procedure GetTransitionsOn( cChar : char; oStateList : TList)
16209: Procedure GetVisibleWindows( List : TStringList)
16210: Procedure GoBegin
16211: Procedure GotoCurrent( DataSet : TCustomClientDataSet)
16212: Procedure GotoCurrent( Table : TTable)
16213: procedure GotoEnd1Click(Sender: TObject);
16214: Procedure GotoNearest
16215: Procedure GradientFillCanvas(const ACanvas:TCanvas;const AStartCol,AEndCol:TColor;const ARect:TRect;const
Direction: TGradientDirection)
16216: Procedure HandleException( E : Exception; var Handled : Boolean)
16217: procedure HANDLEMESSAGE
16218: procedure HandleNeeded;
16219: Procedure Head( AURL : string)
16220: Procedure Help( var AHelpContents : TStringList; ACommand : string)
16221: Procedure HexToBinary( Stream : TStream)
16222: procedure HexToBinary(Stream:TStream)
16223: Procedure HideDragImage
16224: Procedure HideFormCaption( FormHandle : THandle; Hide : Boolean)
16225: Procedure HideTraybar
16226: Procedure HideWindowForSeconds(secs: integer); { //3 seconds}
16227: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm); { //3 seconds}
16228: Procedure HookOSExceptions
16229: Procedure HookSignal( RtlSigNum : Integer)
16230: Procedure HSLToRGB( const H, S, L : Single; out R, G, B : Single);
16231: Procedure HTMLSyntax1Click( Sender : TObject)
16232: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
16233: Procedure IFPS3ClassesPlugin1ExecImport( Sender : TObject; Exec : TPSExec; x : TPSRuntimeClassImporter)
16234: Procedure ImportfromClipboard1Click( Sender : TObject)
16235: Procedure ImportfromClipboard2Click( Sender : TObject)
16236: Procedure IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Integer)
16237: procedure Incb(var x: byte);
16238: Procedure IncludeClick( Sender : TObject)
16239: Procedure IncludeOFF; //preprocessing
16240: Procedure IncludeON;
16241: procedure InfoClick(Sender: TObject);
16242: Procedure InitAltRecBuffers( CheckModified : Boolean)
16243: Procedure InitContext( Request : TWebRequest; Response : TWebResponse)
16244: Procedure InitContext(WebModuleList:TAbstractWebModuleList;Request:TWebRequest;Response:TWebResponse)
16245: Procedure InitData( ASource : TDataSet)
16246: Procedure InitDelta( ADelta : TPacketDataSet);
16247: Procedure InitDelta1( const ADelta : OleVariant);
16248: Procedure InitErrorPacket( E : EUpdateError; Response : TResolverResponse)
16249: Procedure Initialize
16250: procedure InitializePackage(Module: HMODULE)
16251: Procedure INITIATEACTION
16252: Procedure initHexString(var hexn: THexString); //THexArray', 'array[0..15] of char;
16253: Procedure InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet)
16254: Procedure InitModule( AModule : TComponent)
16255: Procedure InitStdConvs
16256: Procedure InitTreeData( Tree : TUpdateTree)
16257: Procedure INSERT
16258: Procedure Insert( Index : Integer; AClass : TClass)
16259: Procedure Insert( Index : Integer; AComponent : TComponent)
16260: Procedure Insert( Index : Integer; AObject : TObject)

```

```

16261: Procedure Insert( Index : Integer; const S : WideString)
16262: Procedure Insert( Index : Integer; Image, Mask : TBitmap)
16263: Procedure Insert(Index: Integer; const S: string);
16264: procedure Insert(Index: Integer; S: string);
16265: procedure Insert(s: AnyString; var s2: AnyString; iPos: Longint);
16266: procedure InsertComponent(AComponent: TComponent)
16267: procedure InsertControl(AControl: TControl);
16268: Procedure InsertIcon( Index : Integer; Image : TIcon)
16269: Procedure InsertMasked( Index : Integer; Image : TBitmap; MaskColor : TColor)
16270: Procedure InsertObject( Index : Integer; const S : WideString; AObject : TObject)
16271: procedure InsertObject(Index:Integer;S:String;AObject:TObject)
16272: Procedure Int16ToBytes( Value : SmallInt; Bytes : array of byte)
16273: Procedure Int32ToBytes( Value : Integer; Bytes : array of byte)
16274: Procedure Int64ToBytes( Value : Int64; Bytes : array of byte)
16275: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
16276: Procedure InternalBeforeResolve( Tree : TUpdateTree)
16277: Procedure InvalidateModuleCache
16278: Procedure InvalidateTitles
16279: Procedure InvalidateDayError( const AYear, ADayOfYear : Word)
16280: Procedure InvalidateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
16281: Procedure InvalidateTimeError(const AYear, AMth, ADay, AHour, AMin, ASec, AMilSec: Word; const
ABaseDate: TDateTime)
16282: Procedure InvalidateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
16283: Procedure InvalidateDayOfWeekInMonthError( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word)
16284: procedure JavaSyntax1Click(Sender: TObject);
16285: Procedure JclLocalesInfoList( const Strings : TStrings; InfoType : Integer)
16286: Procedure KillDataChannel
16287: Procedure Largefont1Click( Sender : TObject)
16288: Procedure LAST
16289: Procedure LaunchCpl( FileName : string)
16290: Procedure Launch( const AFile : string)
16291: Procedure LaunchFile( const AFile : string)
16292: Procedure LetFileList( FileList: TStringList; apath: string);
16293: Procedure lineToNumber( xmemo : String; met : boolean)
16294: Procedure ListViewCustomDrawItem(Sender:TCustListView;Item:TListItem;State:TCustDrawState;var
DefaultDraw:Bool)
16295: Procedure ListViewCustomDrawSubItem( Sender : TCustomListView; Item : TListItem; SubItem : Integer; State
: TCustomDrawState; var DefaultDraw : Boolean)
16296: Procedure ListViewData( Sender : TObject; Item : TListItem)
16297: Procedure ListViewDataFind(Sender:TObject; Find : TItemFind; const FindString : String; const FindPosition
: TPoint; FindData:Pointer; StartIndex:Integer;Direction:TSearchDirection;Wrap:Boolean;var Index: Integer)
16298: Procedure ListViewDataHint( Sender : TObject; StartIndex, EndIndex : Integer)
16299: Procedure ListViewDb1Click( Sender : TObject)
16300: Procedure ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
16301: Procedure ListDLLExports(const FileName: string; List: TStrings);
16302: Procedure Load( const WSDLFileName : WideString; Stream : TMemoryStream)
16303: procedure LoadBytecode1Click(Sender: TObject);
16304: procedure LoadFileFromResource(const FileName: string; ms: TMemoryStream);
16305: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HP)
16306: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
16307: Procedure LoadFromFile( AFileName : string)
16308: Procedure LoadFromFile( const AFileName : string; const AHeadersOnly : Boolean)
16309: Procedure LoadFromFile( const FileName : string)
16310: Procedure LOADFROMFILE( const FILENAME : String; BLOBTYPE : TBLOBTYPE)
16311: Procedure LoadFromFile( const FileName : string; DataType : TDataType)
16312: Procedure LoadFromFile( const FileName : WideString)
16313: Procedure LoadFromFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
16314: Procedure LoadFromFile(const AFileName: string)
16315: procedure LoadFromFile(FileName: string);
16316: procedure LoadFromFile(FileName:String)
16317: Procedure LoadFromResourceID( Instance : THandle; ResID : Integer)
16318: Procedure LoadFromResourceName( Instance : THandle; const ResName : String)
16319: Procedure LoadFromStream( AStream : TStream; const AHeadersOnly : Boolean)
16320: Procedure LoadFromStream( const Stream : TStream)
16321: Procedure LoadFromStream( S : TStream)
16322: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
16323: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
16324: Procedure LoadFromStream( Stream : TStream)
16325: Procedure LOADFROMSTREAM( STREAM : TSTREAM; BLOBTYPE : TBLOBTYPE)
16326: Procedure LoadFromStream( Stream : TStream; DataType : TDataType)
16327: procedure LoadFromStream(Stream: TStream);
16328: Procedure LoadFromStream1( Stream : TSeekableStream; const FormatExt : string);
16329: Procedure LoadFromStream2( Stream : TStream; const FormatExt : string);
16330: Procedure LoadFromStrings( AStrings : TStrings; const MimeSeparator : Char)
16331: Procedure LoadLastFile1Click( Sender : TObject)
16332: { LoadIcoToImage loads two icons from resource named NameRes,
16333: into two image lists ALarge and ASmall}
16334: Procedure LoadIcoToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes: string);
16335: Procedure LoadMemo
16336: Procedure LoadParamsFromIniFile( FFileName : WideString)
16337: Procedure Lock
16338: Procedure Login
16339: Procedure MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
16340: Procedure MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
16341: Procedure MakeCaseInsensitive
16342: Procedure MakeDeterministic( var bChanged : boolean)
16343: Procedure MakeGrayPal( var Palette, ColorCount : Integer)
16344: // type TVolumeLevel = 0..127; , saveFilePath as C:\MyFile.wav
16345: Procedure MakeSound(Frequency{Hz}, Duration{mSec}: Int; Volume: TVolumeLevel; savefilePath: string);

```

```

16346: Procedure MakeComplexSound(N:integer;freqlist:TStrings;Duration{mSec}:Int;pinknoise:bool;Volume:Byte);
16347: Procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
16348: Procedure SetRectComplexFormatStr( const S : string)
16349: Procedure SetPolarComplexFormatStr( const S : string)
16350: Procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
16351: Procedure MakeVisible
16352: Procedure MakeVisible( PartialOK : Boolean)
16353: Procedure ManualClick( Sender : TObject)
16354: Procedure MarkReachable
16355: Procedure maXbox; //shows the exe version data in a win box
16356: Procedure MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
16357: Procedure Memo1Change( Sender : TObject)
16358: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Int;var
Action:TSynReplaceAction)
16359: Procedure Memo1SpecialLineColors(Sender:TObject;Line:Integer; var Special:Boolean;var FG,BG: TColor)
16360: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
16361: procedure Memory1Click(Sender: TObject);
16362: Procedure MERGE( MENU : TMAINMENU)
16363: Procedure MergeChangeLog
16364: procedure MINIMIZE
16365: Procedure MinimizeMaxbox;
16366: Procedure MkDir(const s: string)
16367: Procedure mnuPrintFont1Click( Sender : TObject)
16368: procedure ModalStarted
16369: Procedure Modified
16370: Procedure ModifyAlias( Name : string; List : TStrings)
16371: Procedure ModifyDriver( Name : string; List : TStrings)
16372: Procedure MomentSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4, Skew,Kurtosis:Extended)
16373: Procedure MouseToCell( X, Y : Integer; var ACol, ARow : Longint)
16374: Procedure Move( CurIndex, NewIndex : Integer)
16375: procedure Move(CurIndex, NewIndex: Integer);
16376: procedure Move2(const Source: TByteArray; var Dest: TByteArray; Count: Integer)
16377: Procedure MoveChars(const ASource: String; ASourceStart:int;var ADest: String; ADestStart,ALen:integer)
16378: Procedure moveCube( o : TMyLabel)
16379: Procedure MoveTo( Destination : LongInt; AttachMode : TAttachMode)
16380: procedure MoveTo(X, Y: Integer);
16381: procedure MoveWindowOrg(DC: HDC; DX, DY: Integer);
16382: Procedure MovePoint(var x,y:Extended; const angle:Extended);
16383: Procedure Multiply( Multiplier1, Multiplier2 : TMyBigInt);
16384: Procedure Multiply1( Multiplier1 : TMyBigInt; Multiplier2 : Integer);
16385: Procedure MsgAbout(Handle:Int;const Msg,Caption:string;const IcoName:string = 'MAINICON';Flags:DWORD=MB_OK);
16386: Procedure mxButton(x,y,width,height,top,left,ahandle: integer);
16387: Procedure New( Width, Height : Integer; PixFormat : TPixelFormat)
16388: procedure New(P: PChar)
16389: procedure New1Click(Sender: TObject);
16390: procedure NewInstance1Click(Sender: TObject);
16391: Procedure NEXT
16392: Procedure NextMonth
16393: Procedure Noop
16394: Procedure NormalizePath( var APath : string)
16395: procedure ObjectBinaryToText(Input, Output: TStream)
16396: procedure ObjectBinaryToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
16397: procedure ObjectResourceToText(Input, Output: TStream)
16398: procedure ObjectResourceToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
16399: procedure ObjectTextToBinary(Input, Output: TStream)
16400: procedure ObjectTextToBinary1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
16401: procedure ObjectTextToResource(Input, Output: TStream)
16402: procedure ObjectTextToResource1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
16403: Procedure Open( const Name, Address, Service : string; Port : Word; Block : Boolean)
16404: Procedure Open( const UserID : WideString; const Password : WideString);
16405: Procedure Open;
16406: Procedure open1Click( Sender : TObject)
16407: Procedure OpenCdDrive
16408: Procedure OpenCloseCdDrive( OpenMode : Boolean; Drive : Char)
16409: Procedure OpenCurrent
16410: Procedure OpenFile(vfilenamepath: string)
16411: Procedure OpenDirectory1Click( Sender : TObject)
16412: Procedure OpenIndexFile( const IndexName : string)
16413: Procedure OpenSchema(const Schema:TSchemaInf;const Restricts:OleVar;const
SchemaID:OleVariant;DataSet:TADODataSet)
16414: Procedure OpenWriteBuffer( const AThreshold : Integer)
16415: Procedure OptimizeMem
16416: Procedure Options1( AURL : string);
16417: Procedure OutputDebugString(lpOutputString : PChar)
16418: Procedure PackBuffer
16419: Procedure Paint
16420: Procedure PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch : Boolean)
16421: Procedure PaintToTBitmap( Target : TBitmap)
16422: Procedure PaletteChanged
16423: Procedure ParentBiDiModeChanged
16424: Procedure PARENTBIDIMODECHANGED( ACONTROL : TOBJECT)
16425: Procedure PasteFromClipboard;
16426: Procedure PasteImage( Source : TLinearBitmap; X, Y : Integer)
16427: Procedure PathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
16428: Procedure PerformEraseBackground(Control: TControl; DC: HDC);
16429: Procedure PError( Text : string)
16430: procedure Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
16431: procedure Pie(X1:Integer;Y1:Integer;X2:Integer;Y2:Integer;X3:Integer;Y3:Integer;X4:Integer;Y4:Integer);
16432: Procedure Play( FromFrame, ToFrame : Word; Count : Integer)

```

```

16433: procedure playmp3(mpath: string);
16434: Procedure PlayMP3Click( Sender : TObject)
16435: Procedure PointCopy( var Dest : TPoint; const Source : TPoint)
16436: Procedure PointMove( var P : TPoint; const DeltaX, DeltaY : Integer)
16437: procedure PolyBezier(const Points: array of TPoint);
16438: procedure PolyBezierTo(const Points: array of TPoint);
16439: procedure Polygon(const Points: array of TPoint);
16440: procedure Polyline(const Points: array of TPoint);
16441: Procedure Pop
16442: Procedure POPULATEOLE2MENU(SHAREDMENU: HMENU;GROUPS: array of INT; var WIDTHS : array of LONGINT)
16443: Procedure PopulationVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
16444: Procedure POPUP( X, Y : INTEGER)
16445: Procedure PopUpURL(URL : WideString);
16446: Procedure POST
16447: Procedure Post4( AURL : string; const ASource : TStrings; const AResponseContent : TStream);
16448: Procedure Post5( AURL : string; const ASource, AResponseContent : TStream);
16449: Procedure Post6( AURL : string; const ASource : TIdMultiPartFormDataStream; AResponseContent : TStream);
16450: Procedure PostUser( const Email, FirstName, LastName : WideString)
16451: Procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
16452: procedure Pred(X: int64);
16453: Procedure Prepare
16454: Procedure PrepareStatement
16455: Procedure PreProcessXML( AList : TStrings)
16456: Procedure PreventDestruction
16457: Procedure Print( const Caption : string)
16458: procedure PrintBitmap(aGraphic: TGraphic; Title: string);
16459: procedure printf(const format: String; const args: array of const);
16460: Procedure PrintList(Value: TStringList);
16461: Procedure PrintImage(aValue:TBitmap;Style:TBitmapStyle);//TBitmapStyle=(bsNormal,bsCentered,bsStretched)
16462: Procedure PrintoutClick( Sender : TObject)
16463: Procedure ProcessHeaders
16464: Procedure PROCESSMENUCHAR( var MESSAGE : TWMMENUCHAR)
16465: Procedure ProcessMessage( AMsg : TIdMessage; AHeaderOnly : Boolean);
16466: Procedure ProcessMessage1( AMsg : TIdMessage; const AStream : TStream; AHeaderOnly : Boolean);
16467: Procedure ProcessMessage2( AMsg : TIdMessage; const AFilename : string; AHeaderOnly : Boolean);
16468: Procedure ProcessMessagesOFF; //application.processmessages
16469: Procedure ProcessMessagesON;
16470: Procedure ProcessPath(const EditText:string;var Drive:Char; var DirPart:string; var FilePart : string)
16471: Procedure ProcessPath1(const EditText:string;var Drive:Char; var DirPart:string; var FilePart:string);
16472: Procedure ProcList Size is: 3797 /1415
16473: Procedure procMessClick( Sender : TObject)
16474: Procedure PSScriptCompile( Sender : TPSScript)
16475: Procedure PSScriptExecute( Sender : TPSScript)
16476: Procedure PSScriptLine( Sender : TObject)
16477: Procedure Push( ABoundary : string)
16478: procedure PushItem(AItem: Pointer)
16479: Procedure Put2( AURL : string; const ASource, AResponseContent : TStream);
16480: Procedure Put2( const ASourceFile : string; const ADestFile : string; const AAppend : boolean);
16481: procedure PutLinuxLines(const Value: string)
16482: Procedure Quit
16483: Procedure RaiseConversionError( const AText : string);
16484: Procedure RaiseConversionError1( const AText : string; const AArgs : array of const);
16485: Procedure RaiseConversionRegError( AFamily : TConvFamily; const ADescription : string)
16486: procedure RaiseException(Ex: TIFException; Param: String);
16487: Procedure RaiseExceptionForLastCmdResult;
16488: procedure RaiseLastException;
16489: procedure RaiseException2;
16490: Procedure RaiseLastOSError
16491: Procedure RaiseLastWin32;
16492: procedure RaiseLastWin32Error)
16493: Procedure RaiseListError( const ATemplate : string; const AData : array of const)
16494: RandomFillStream( Stream : TMemoryStream)
16495: procedure randomize;
16496: Procedure Rasterize( Rasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect)
16497: Procedure RCS
16498: Procedure Read( Socket : TSocket)
16499: Procedure ReadBlobData
16500: procedure ReadBuffer(Buffer:String;Count:LongInt)
16501: procedure ReadOnlyClick(Sender: TObject);
16502: Procedure ReadSection( const Section : string; Strings : TStrings)
16503: Procedure ReadSections( Strings : TStrings)
16504: Procedure ReadSections( Strings : TStrings);
16505: Procedure ReadSections1( const Section : string; Strings : TStrings);
16506: Procedure ReadSectionValues( const Section : string; Strings : TStrings)
16507: Procedure ReadStream( AStream : TStream; AByteCount:LongInt; const AReadUntilDisconnect : boolean)
16508: Procedure ReadStrings( ADest : TStrings; AReadLinesCount : Integer)
16509: Procedure ReadVersion2(aFileName: STRING; aVersion : TStrings);
16510: Function ReadVersion(aFileName: STRING; aVersion : TStrings): boolean;
16511: Procedure Realign;
16512: procedure Rectangle(X1, Y1, X2, Y2: Integer);
16513: Procedure Rectangle1( const Rect : TRect);
16514: Procedure RectCopy( var Dest : TRect; const Source : TRect)
16515: Procedure RectFitToScreen( var R : TRect)
16516: Procedure RectGrow( var R : TRect; const Delta : Integer)
16517: Procedure RectGrowX( var R : TRect; const Delta : Integer)
16518: Procedure RectGrowY( var R : TRect; const Delta : Integer)
16519: Procedure RectMove( var R : TRect; const DeltaX, DeltaY : Integer)
16520: Procedure RectMoveTo( var R : TRect; const X, Y : Integer)
16521: Procedure RectNormalize( var R : TRect)

```



```

16522: // TFileCallbackProcedure = procedure(filename:string);
16523: Procedure RecurseDirectory(Dir: String; IncludeSubs: boolean; callback: TFileCallbackProcedure);
16524: Procedure RecurseDirectory2(Dir: String; IncludeSubs : boolean);
16525: Procedure RedirectTransition(oOldState:TniRegularExpressionState; oNewState : TniRegularExpressionState)
16526: Procedure Refresh;
16527: Procedure RefreshData( Options : TFetchOptions)
16528: Procedure REFRESHLOOKUPLIST
16529: Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
16530: Procedure RegisterChanges( Value : TChangeLink)
16531: Procedure RegisterConversionFormat( const AExtension : string; AConversionClass : TConversionClass)
16532: Procedure RegisterFileFormat( const AExtension, ADescription : string; AGraphicClass : TGraphicClass)
16533: Procedure RegisterFileFormat(Extension,AppID:string;Description:string;Executable:string;IconIndex:Int)
16534: Procedure ReInitialize( ADelay : Cardinal)
16535: Procedure RELEASE
16536: Procedure Remove( const AByteCount : integer)
16537: Procedure REMOVE( FIELD : TFIELD)
16538: Procedure REMOVE( ITEM : TMenuItem)
16539: Procedure REMOVE( POPUP : TPOPMENU)
16540: Procedure RemoveAllPasswords
16541: procedure RemoveComponent(AComponent:TComponent)
16542: Procedure RemoveDir( const ADirName : string)
16543: Procedure RemoveLambdaTransitions( var bChanged : boolean)
16544: Procedure REMOVEPARAM( VALUE : TPARAM)
16545: Procedure RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharSet);
16546: Procedure RemoveTransitionTo( oState : TniRegularExpressionState);
16547: Procedure Rename( const ASourceFile, ADestFile : string)
16548: Procedure Rename( const FileName : string; Reload : Boolean)
16549: Procedure RenameTable( const NewTableName : string)
16550: Procedure Replace( Index : Integer; Image, Mask : TBitmap)
16551: Procedure ReplaceClick( Sender : TObject)
16552: Procedure ReplaceDate( var DateTime : TDateTime; NewDate : TDateTime)
16553: procedure ReplaceDate(var DateTime: TDateTime; const NewDate: TDateTime)
16554: Procedure ReplaceIcon( Index : Integer; Image : TIcon)
16555: Procedure ReplaceMasked( Index : Integer; NewImage : TBitmap; MaskColor : TColor)
16556: Procedure ReplaceTime( var DateTime : TDateTime; NewTime : TDateTime)
16557: procedure ReplaceTime(var DateTime: TDateTime; const NewTime: TDateTime);
16558: Procedure Requery( Options : TExecuteOptions)
16559: Procedure Reset
16560: Procedure ResetClick( Sender : TObject)
16561: Procedure ResizeCanvas( XSiz, YSiz, XPos, YPos : Integer; Color : TColor)
16562: procedure ResourceExploreClick(Sender: TObject);
16563: Procedure RestoreContents
16564: Procedure RestoreDefaults
16565: Procedure RestoreOtherInstance( MainFormClassName, MainFormCaption : string)
16566: Procedure RetrieveHeaders
16567: Procedure RevertRecord
16568: Procedure RGBAToBGRA( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal)
16569: Procedure RGBToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
16570: Procedure RGBToBGR1( const R, G, B, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
16571: Procedure RGBToHSL( const R, G, B : Single; out H, S, L : Single);
16572: Procedure RGBToHSL1( const RGB : TColor32; out H, S, L : Single);
16573: Procedure RGBToHSV( r, g, b : Integer; var h, s, v : Integer)
16574: Procedure RleCompress2( Stream : TStream)
16575: Procedure RleDecompress2( Stream : TStream)
16576: Procedure Rmdir(const S: string)
16577: Procedure Rollback
16578: Procedure Rollback( TransDesc : TTransactionDesc)
16579: Procedure RollbackFreeAndNil( var Transaction : TDBXTransaction)
16580: Procedure RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction)
16581: Procedure RollbackTrans
16582: procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer);
16583: Procedure RoundToAllocGranularity64( var Value : Int64; Up : Boolean)
16584: Procedure RoundToAllocGranularityPtr( var Value : Pointer; Up : Boolean)
16585: Procedure RunDll32Internal( Wnd : HWND; const DLLName, FuncName, CmdLine : string; CmdShow : Integer)
16586: Procedure S_AddMessageToStrings( AMessages : TStrings; AMsg : string)
16587: Procedure S_EBox( const AText : string)
16588: Procedure S_GetEncryptionKeys(DateTime1,DateTime2:TDateTime;var StartKey:int;var MultKey:integer;var
AddKey:int
16589: Procedure S_IBox( const AText : string)
16590: Procedure S_ReplaceChar( var cStr : string; cOldChr, cNewChr : char)
16591: Procedure S_ReplaceStringInFile( AFileName : string; ASearchString, AReplaceString : string)
16592: Procedure S_TokenInit( cBuffer : PChar; const cDelimiters : string)
16593: Procedure SampleVarianceAndMean
16594: ( const X : TDynFloatArray; var Variance, Mean : Float)
16595: Procedure Save2Click( Sender : TObject)
16596: Procedure Saveas3Click( Sender : TObject)
16597: Procedure Savebefore1Click( Sender : TObject)
16598: Procedure SaveBytesToFile(const Data: TBytes; const FileName: string);
16599: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
16600: Procedure SaveConfigFile
16601: Procedure SaveOutput1Click( Sender : TObject)
16602: procedure SaveScreenshotClick(Sender: TObject);
16603: Procedure SaveLn(pathname, content: string); //SaveIn(exepath+'mysaveIntest.txt', memo2.text);
16604: Procedure SaveToClipboardFormat( var AFormat : Word; var AData : THandle; var APalette : HPALETTE)
16605: Procedure SaveToClipboardFormat( var Format : Word; var Data : THandle; var APalette : HPALETTE)
16606: Procedure SaveToFile( AFileName : string)
16607: Procedure SAVETOFILE( const FILENAME : String)
16608: Procedure SaveToFile( const FileName : WideString)
16609: Procedure SaveToFile( const FileName : WideString; Format : TPersistFormat)

```

```

16610: Procedure SaveToFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
16611: procedure SaveToFile(FileName: string);
16612: procedure SaveToFile(FileName:String)
16613: Procedure SaveToStream( AStream : TStream; const AHeadersOnly : Boolean)
16614: Procedure SaveToStream( OutStream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
16615: Procedure SaveToStream( S : TStream)
16616: Procedure SaveToStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
16617: Procedure SaveToStream( Stream : TStream)
16618: Procedure SaveToStream( Stream : TStream; Format : TDataPacketFormat)
16619: procedure SaveToStream(Stream: TStream);
16620: procedure SaveToStream(Stream:TStream)
16621: Procedure SaveToStream1( Stream : TSeekableStream; const FormatExt : string);
16622: Procedure SaveToStream2( Stream : TStream; const FormatExt : string);
16623: Procedure SaveToStrings( AStrings : TStrings; const MimeSeparator : Char)
16624: procedure Say(const sText: string)
16625: Procedure SBytecode1Click( Sender : TObject)
16626: Procedure ScaleImage( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double)
16627: procedure ScriptExplorer1Click(Sender: TObject);
16628: Procedure Scroll( Distance : Integer)
16629: Procedure Scroll( DX, DY : Integer)
16630: procedure ScrollBy(DeltaX, DeltaY: Integer);
16631: procedure SCROLLINVIEW(ACONTROL:TCONTROL)
16632: Procedure ScrollTabs( Delta : Integer)
16633: Procedure Search1Click( Sender : TObject)
16634: procedure SearchAndOpenDoc(vfilenamepath: string)
16635: procedure SearchAndOpenFile(vfilenamepath: string)
16636: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string)
16637: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
16638: Procedure SearchNext1Click( Sender : TObject)
16639: Procedure Select( Node : TTreeNode; ShiftState : TShiftState);
16640: Procedure Select1( const Nodes : array of TTreeNode);
16641: Procedure Select2( Nodes : TList);
16642: Procedure SelectNext( Direction : Boolean)
16643: Procedure SelectNextPage( GoForward : Boolean; CheckTabVisible : Boolean)
16644: Procedure SelfTestPEM //unit uPSI_cPEM
16645: Procedure Send( AMsg : TIdMessage)
16646: //config forst in const MAILINIFILE = 'maildef.ini';
16647: //ex.: SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox',
16648: Procedure SendEmail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
16649: Procedure SendMail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
16650: Procedure SendMsg( AMsg : TIdMessage; const AHeadersOnly : Boolean)
16651: Procedure SendMsg(AMsg: TIdMessage; const AHeadersOnly: Boolean = False)
16652: Procedure SendResponse
16653: Procedure SendStream( AStream : TStream)
16654: Procedure Set8087CW( NewCW : Word)
16655: Procedure SetAll( One, Two, Three, Four : Byte)
16656: Procedure SetAltRecBuffers( Old, New, Cur : PChar)
16657: Procedure SetAppDispatcher( const ADispatcher : TComponent)
16658: procedure SetArrayLength;
16659: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer); //2 dimension
16660: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
16661: Procedure SetAsHandle( Format : Word; Value : THandle)
16662: procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer)
16663: procedure SetCaptureControl(Control: TControl);
16664: Procedure SetColumnAttributes
16665: Procedure SetCookieField(Values:TStrings;const ADomain,APath:string;AExpires:TDateTime;ASecure:Boolean)
16666: Procedure SetCustomHeader( const Name, Value : string)
16667: Procedure SetExprParams(const Text:Widestring;Opts:TFilterOpts;ParserOpts:TParserOpts;const
FieldNames:Widestring)
16668: Procedure SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
16669: Procedure SetFocus
16670: procedure SetFocus; virtual;
16671: Procedure SetInitialState
16672: Procedure SetKey
16673: procedure SetLastError(ErrorCode: Integer)
16674: procedure SetLength;
16675: Procedure SetLineBreakStyle( var T : Text; Style : TTextLineBreakStyle)
16676: Procedure SETOLE2MENUHANDLE( HANDLE : HMENU)
16677: Procedure SetParams( ADataset : TDataset; UpdateKind : TUpdateKind);
16678: procedure SETPARAMS(APOSITION,AMIN,AMAX:INTEGER)
16679: Procedure SetParams1( UpdateKind : TUpdateKind);
16680: Procedure SetPassword( const Password : string)
16681: Procedure SetPointer( Ptr : Pointer; Size : Longint)
16682: Procedure SetPrimalityTest( const Method : TPrimalityTestMethod)
16683: Procedure SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
16684: Procedure SetProvider( Provider : TComponent)
16685: Procedure SetProxy( const Proxy : string)
16686: Procedure SetPSResult( var PSResult : TPSResult; Value : TObject)
16687: Procedure SetRange( const StartValues, EndValues : array of const)
16688: Procedure SetRangeEnd
16689: Procedure SetRate( const aPercent, aYear : integer)
16690: procedure SetRate(const aPercent, aYear: integer)
16691: Procedure Set_ReportMemoryLeaksOnShutdown(ab0: boolean)
16692: Procedure SetSafeCallExceptionHandler( Msg : string)
16693: procedure SETSELTEXTBUF(BUFFER:PCHAR)
16694: Procedure SetSize( AWidth, AHeight : Integer)
16695: procedure SetSize(NewSize:LongInt)
16696: procedure SetString(var s: string; buffer: PChar; len: Integer)
16697: Procedure SetStrings( List : TStrings)

```

```

16698: Procedure SetText( Text : PwideChar)
16699: procedure SetText(Text: PChar);
16700: Procedure SetTextBuf( Buffer : PChar)
16701: procedure SETTEXTBUF(BUFFER:PCHAR)
16702: Procedure SetTick( Value : Integer)
16703: Procedure SetTimeout( ATimeOut : Integer)
16704: Procedure SetTraceEvent( Event : TDBXTraceEvent)
16705: Procedure SetUserName( const UserName : string)
16706: Procedure SetWallpaper( Path : string);
16707: procedure ShellStyle1Click(Sender: TObject);
16708: Procedure SHORTCUTTOKEY( SHORTCUT : TSHORTCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
16709: Procedure ShowFileProperties( const FileName : string)
16710: Procedure ShowInclude1Click( Sender : TObject)
16711: Procedure ShowInterfaces1Click( Sender : TObject)
16712: Procedure ShowLastException1Click( Sender : TObject)
16713: Procedure ShowMessage( const Msg : string)
16714: Procedure ShowMessageBig(const aText : string);
16715: Procedure ShowMessageBig2(const aText : string; aautosize: boolean);
16716: Procedure ShowMessageBig3(const aText : string; fsize: byte; aautosize: boolean);
16717: Procedure MsgBig(const aText : string); //alias
16718: procedure showmessage(mytext: string);
16719: Procedure ShowMessageFmt( const Msg : string; Params : array of const)
16720: procedure ShowMessageFmt(const Msg: string; Params: array of const))
16721: Procedure ShowMessagePos( const Msg : string; X, Y : Integer)
16722: procedure ShowMessagePos(const Msg: string; X: Integer; Y: Integer))
16723: Procedure ShowSearchDialog( const Directory : string)
16724: Procedure ShowSpecChars1Click( Sender : TObject)
16725: Procedure ShowBitmap(bmap: TBitmap); //draw in a form!
16726: Procedure ShredFile( const FileName : string; Times : Integer)
16727: procedure Shuffle(vQ: TStringList);
16728: Procedure ShuffleList( var List : array of Integer; Count : Integer)
16729: Procedure SimulateKeystroke( Key : byte; Shift : TShiftState)
16730: Procedure SinCos( const Theta : Extended; var Sin, Cos : Extended)
16731: Procedure SinCosE( X : Extended; out Sin, Cos : Extended)
16732: Procedure Site( const ACommand : string)
16733: Procedure SkipEOL
16734: Procedure Sleep( ATime : cardinal)
16735: Procedure Sleep( milliseconds : Cardinal)
16736: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
16737: Procedure Slinenumber1Click( Sender : TObject)
16738: Procedure Sort
16739: Procedure SortColorArray(ColorArray:TColorArray;L,R:Int;SortType:TColorArraySortType;Reverse:Bool)
16740: procedure Speak(const sText: string) //async like voice
16741: procedure Speak2(const sText: string) //sync
16742: procedure Split(Str: string; SubStr: string; List: TStringList);
16743: Procedure SplitNameValue( const Line : string; var Name, Value : string)
16744: Procedure SplitColumns( const AData : String; AStrings : TStringList; const ADelim : String)
16745: Procedure SplitColumnsNoTrim( const AData : String; AStrings : TStringList; const ADelim : String)
16746: Procedure SplitLines( AData : PChar; ADataSize : Integer; AStrings : TStringList)
16747: Procedure SplitString( const AStr, AToken : String; var VLeft, VRight : String)
16748: procedure SQLSyntax1Click(Sender: TObject);
16749: Procedure SRand( Seed : RNG_IntType)
16750: Procedure Start
16751: Procedure StartCount( var Counter : TJclCounter; const Compensate : Boolean)
16752: procedure StartFileFinder3(spath,aext,searchstr: string; arecursiv: boolean; reslist: TStringlist);
16753: Procedure StartTransaction( TransDesc : TTransactionDesc)
16754: Procedure Status( var AStatusList : TStringList)
16755: Procedure StatusBar1Db1Click( Sender : TObject)
16756: Procedure StepInto1Click( Sender : TObject)
16757: Procedure StepIt
16758: Procedure StepOut1Click( Sender : TObject)
16759: Procedure Stop
16760: procedure stopmp3;
16761: procedure StartWeb(aurl: string);
16762: Procedure Str(aint: integer; astr: string); //of system
16763: Procedure StrDispose( Str : PChar)
16764: procedure StrDispose(Str: PChar)
16765: Procedure StrReplace(var Str: String; Old, New: String);
16766: Procedure StretchDIBits( DC : THandle; const Dest : TRect; HalftoneStretch : Boolean)
16767: procedure StretchDraw(const Rect: TRect; Graphic: TGraphic);
16768: Procedure StringToBytes( Value : String; Bytes : array of byte)
16769: Procedure StrSet(c : Char; I : Integer; var s : String);
16770: Procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStringList);
16771: Procedure StructureMount( APath : String)
16772: procedure STYLECHANGED(SENDER:TObJECT)
16773: Procedure Subselect( Node : TTreeNode; Validate : Boolean)
16774: procedure Succ(X: int64);
16775: Procedure SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares : Extended)
16776: procedure SwapChar(var X,Y: char); //swapX follows
16777: Procedure SwapFloats( var X, Y : Float)
16778: procedure SwapGrid(grd: TStringGrid);
16779: Procedure SwapOrd( var I, J : Byte);
16780: Procedure SwapOrd( var X, Y : Integer)
16781: Procedure SwapOrd1( var I, J : Shortint);
16782: Procedure SwapOrd2( var I, J : Smallint);
16783: Procedure SwapOrd3( var I, J : Word);
16784: Procedure SwapOrd4( var I, J : Integer);
16785: Procedure SwapOrd5( var I, J : Cardinal);
16786: Procedure SwapOrd6( var I, J : Int64);

```

```

16787: Procedure SymetricCompareFiles(const plaintext, replaintext: string)
16788: Procedure Synchronizel( Method : TMethod);
16789: procedure SyntaxCheck1Click(Sender: TObject);
16790: Procedure SysFreeString(const S: WideString); stdcall;
16791: Procedure TakeOver( Other : TLinearBitmap)
16792: Procedure Talkln(const sText: string) //async voice
16793: procedure tbtn6resClick(Sender: TObject);
16794: Procedure tbtnUseCaseClick( Sender : TObject)
16795: procedure TerminalStyle1Click(Sender: TObject);
16796: Procedure Terminate
16797: Procedure texSyntax1Click( Sender : TObject)
16798: procedure TextOut(X, Y: Integer; Text: string);
16799: Procedure TextRect( Rect : TRect; X, Y : Integer; const Text : string);
16800: procedure TextRect(Rect: TRect; X: Integer; Y: Integer; const Text: string);
16801: Procedure TextRect1( var Rect : TRect; var Text : string; TextFormat : TTextFormat);
16802: Procedure TextStart
16803: procedure TILE
16804: Procedure TimeStampToBytes( Value : TBcd; Bytes : array of byte)
16805: Procedure TitleClick( Column : TColumn)
16806: Procedure ToDo
16807: procedure toolbtnTutorialClick(Sender: TObject);
16808: Procedure Trace1( AURL : string; const AResponseContent : TStream);
16809: Procedure TransferMode( ATransferMode : TIdFTPTransferMode)
16810: Procedure Truncate
16811: procedure Tutorial101Click(Sender: TObject);
16812: procedure Tutorial10Statistics1Click(Sender: TObject);
16813: procedure Tutorial11Forms1Click(Sender: TObject);
16814: procedure Tutorial12SQL1Click(Sender: TObject);
16815: Procedure tutorial1Click( Sender : TObject)
16816: Procedure tutorial21Click( Sender : TObject)
16817: Procedure tutorial31Click( Sender : TObject)
16818: Procedure tutorial4Click( Sender : TObject)
16819: Procedure Tutorial5Click( Sender : TObject)
16820: procedure Tutorial6Click(Sender: TObject);
16821: procedure Tutorial91Click(Sender: TObject);
16822: Procedure UnhookSignal( RtlSigNum : Integer; OnlyIfHooked : Boolean)
16823: procedure UniqueString(var str: AnsiString)
16824: procedure UnloadLoadPackage(Module: HMODULE)
16825: Procedure Unlock
16826: Procedure UNMERGE( MENU : TMAINMENU)
16827: Procedure UnRegisterChanges( Value : TChangeLink)
16828: Procedure UnregisterConversionFamily( const AFamily : TConvFamily)
16829: Procedure UnregisterConversionType( const AType : TConvType)
16830: Procedure UnRegisterProvider( Prov : TCustomProvider)
16831: Procedure UPDATE
16832: Procedure UpdateBatch( AffectRecords : TAffectRecords)
16833: Procedure UPDATECURSORPOS
16834: Procedure UpdateFile
16835: Procedure UpdateItems( FirstIndex, LastIndex : Integer)
16836: Procedure UpdateResponse( AResponse : TWebResponse)
16837: Procedure UpdateScrollBar
16838: Procedure UpdateView1Click( Sender : TObject)
16839: procedure Val(const s: string; var n, z: Integer)
16840: procedure VarArraySet(c: Variant; I: Integer; var s: Variant);
16841: Procedure VarFMTBcdCreate( var ADest : Variant; const ABcd : TBcd);
16842: Procedure VariantAdd( const src : Variant; var dst : Variant)
16843: Procedure VariantAnd( const src : Variant; var dst : Variant)
16844: Procedure VariantArrayRedim( var V : Variant; High : Integer)
16845: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
16846: Procedure VariantClear( var V : Variant)
16847: Procedure VariantCpy( const src : Variant; var dst : Variant)
16848: Procedure VariantDiv( const src : Variant; var dst : Variant)
16849: Procedure VariantMod( const src : Variant; var dst : Variant)
16850: Procedure VariantMul( const src : Variant; var dst : Variant)
16851: Procedure VariantOr( const src : Variant; var dst : Variant)
16852: Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
16853: Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
16854: Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
16855: Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
16856: Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
16857: Procedure VariantShl( const src : Variant; var dst : Variant)
16858: Procedure VariantShr( const src : Variant; var dst : Variant)
16859: Procedure VariantSub( const src : Variant; var dst : Variant)
16860: Procedure VariantXor( const src : Variant; var dst : Variant)
16861: Procedure VarCastError;
16862: Procedure VarCastError1( const ASourceType, ADestType : TVarType);
16863: Procedure VarInvalidOp
16864: Procedure VarInvalidNullOp
16865: Procedure VarOverflowError( const ASourceType, ADestType : TVarType)
16866: Procedure VarRangeCheckError( const ASourceType, ADestType : TVarType)
16867: Procedure VarArrayCreateError
16868: Procedure VarResultCheck( AResult : HRESULT);
16869: Procedure VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);
16870: Procedure HandleConversionException( const ASourceType, ADestType : TVarType)
16871: Function VarTypeAsText( const AType : TVarType) : string
16872: procedure Voice(const sText: string) //async
16873: procedure Voice2(const sText: string) //sync
16874: Procedure WaitMiliSeconds( AMSec : word)
16875: Procedure WideAppend( var dst : WideString; const src : WideString)

```



```

16876: Procedure WideAssign( var dst : WideString; var src : WideString)
16877: Procedure WideDelete( var dst : WideString; index, count : Integer)
16878: Procedure WideFree( var s : WideString)
16879: Procedure WideFromAnsi( var dst : WideString; const src : AnsiString)
16880: Procedure WideFromPChar( var dst : WideString; src : PChar)
16881: Procedure WideInsert( var dst : WideString; const src : WideString; index : Integer)
16882: Procedure WideSetLength( var dst : WideString; len : Integer)
16883: Procedure WideString2Stream( aWideString : WideString; oStream : TStream)
16884: Procedure WideStringToBytes( Value : WideString; Bytes : array of byte)
16885: Procedure WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
16886: Procedure WinInet_HttpGet( const Url: string; Stream:TStream);
16887: Procedure HttpGet( const Url: string; Stream:TStream);
16888: Procedure WordToTwoBytes( AWord : Word; ByteArray : TIdBytes; Index : integer)
16889: Procedure WordWrap1Click( Sender : TObject)
16890: Procedure Write( const AOut : string)
16891: Procedure Write( Socket : TSocket)
16892: procedure Write(S: string);
16893: Procedure WriteBinaryStream( const Section, Name : string; Value : TStream)
16894: Procedure WriteBool( const Section, Ident : string; Value : Boolean)
16895: Procedure WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow : Boolean)
16896: procedure WriteBuffer(Buffer:String;Count:LongInt)
16897: Procedure WriteCardinal( AValue : Cardinal; const AConvert : Boolean)
16898: Procedure WriteChar( AValue : Char)
16899: Procedure WriteDate( const Section, Name : string; Value : TDateTime)
16900: Procedure WriteDateTime( const Section, Name : string; Value : TDateTime)
16901: Procedure WriteFloat( const Section, Name : string; Value : Double)
16902: Procedure WriteHeader( AHeader : TStrings)
16903: Procedure WriteInteger( AValue : Integer; const AConvert : Boolean)
16904: Procedure WriteInteger( const Section, Ident : string; Value : Longint)
16905: Procedure WriteLn( const AOut : string)
16906: procedure Writeln(s: string);
16907: Procedure WriteLog( const FileName, LogLine : string)
16908: Procedure WriteRFCReply( AReply : TIdRFCReply)
16909: Procedure WriteRFCStrings( AStrings : TStrings)
16910: Procedure WriteSmallInt( AValue : SmallInt; const AConvert : Boolean)
16911: Procedure WriteStream(AStream:TStream;const AAll:Bool;const AWriteByteCount:Bool;const ASize:Int)
16912: Procedure WriteString( const Section, Ident, Value : string)
16913: Procedure WriteStrings( AValue : TStrings; const AWriteLinesCount : Boolean)
16914: Procedure WriteTime( const Section, Name : string; Value : TDateTime)
16915: Procedure WriteObjectResourceHeader( ObjStream, Output : TStream)
16916: Procedure Write16bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
16917: Procedure Write32bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
16918: procedure WStrSet(c : AnyString; I : Integer; var s : AnyString);
16919: procedure XMLSyntax1Click(Sender: TObject);
16920: Procedure XOR_Streams2( Dest, Srce : TMemoryStream)
16921: Procedure XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
16922: Procedure ZeroFillStream( Stream : TMemoryStream)
16923: procedure XMLSyntax1Click(Sender: TObject);
16924: Procedure ZeroMemory( Ptr : Pointer; Length : Longint)
16925: procedure(Control: TWinControl; Index: Integer; Rect: TRect; State: Byte)
16926: procedure(Control: TWinControl; Index: Integer; var Height: Integer)
16927: procedure(Sender, Source: TObject; X, Y: Integer; State: TDragState; var Accept: Boolean)
16928: procedure(Sender, Source: TObject; X, Y: Integer)
16929: procedure(Sender, Target: TObject; X, Y: Integer)
16930: procedure(Sender: TObject; ASection, AWidth: Integer)
16931: procedure(Sender: TObject; ScrollCode: TScrollCode; var ScrollPos: Integer)
16932: procedure(Sender: TObject; Shift: TShiftState; X, Y: Integer);
16933: procedure(Sender: TObject; var Action: TCloseAction)
16934: procedure(Sender: TObject; var CanClose: Boolean)
16935: procedure(Sender: TObject; var Key: Char);
16936: ProcedureName ProcedureNames ProcedureParametersCursor @
16937:
16938: *****Now Constructors constructor *****
16939: Size is: 1231 1115 996 628 550 544 501 459 (381)
16940: Attach( VersionInfoData : Pointer; Size : Integer)
16941: constructor Create( ABuckets : TBucketListSizes)
16942: Create( ACallBackWnd : HWND)
16943: Create( AClient : TCustomTaskDialog)
16944: Create( AClient : TIdTelnet)
16945: Create( ACollection : TCollection)
16946: Create( ACollection : TFavoriteLinkItems)
16947: Create( ACollection : TTaskDialogButtons)
16948: Create( AConnection : TIdCustomHTTP)
16949: Create( ACreateSuspended : Boolean)
16950: Create( ADataSet : TCustomSQLDataSet)
16951: CREATE( ADATASET : TDATASET)
16952: Create( Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
16953: Create( AGrid : TCustomDBGrid)
16954: Create( AGrid : TStringGrid; AIndex : Longint)
16955: Create( AHTTP : TIdCustomHTTP)
16956: Create( AListItems : TListItems)
16957: Create( AOnBytesRemoved : TIdBufferBytesRemoved)
16958: Create( AOnBytesRemoved : TIdBufferBytesRemoved)
16959: Create( AOwner : TCommonCalendar)
16960: Create( AOwner : TComponent)
16961: CREATE( AOWNER : TCOMPONENT)
16962: Create( AOwner : TCustomListView)
16963: Create( AOwner : TCustomOutline)
16964: Create( AOwner : TCustomRichEdit)

```

```

16965: Create( AOwner : TCustomRichEdit; AttributeType : TAttributeType)
16966: Create( AOwner : TCustomTreeView)
16967: Create( AOwner : TIdUserManager)
16968: Create( AOwner : TListItems)
16969: Create(
  AOwner:TObject;Handle:hDBICur;CBType:CBType;CBBuf:Ptr;CBBufSize:Int;CallbackEvent:TBDECallbackEvent;
  Chain:Bool)
16970: CREATE( AOWNER : TPERSISTENT)
16971: Create( AOwner : TPersistent)
16972: Create( AOwner : TTable)
16973: Create( AOwner : TTreeNodes)
16974: Create( AOwner : TWinControl; const ClassName : string)
16975: Create( AParent : TIdCustomHTTP)
16976: Create( AParent : TUpdateTree; AResolver : TCustomResolver)
16977: Create( AProvider : TBaseProvider)
16978: Create( AProvider : TCustomProvider);
16979: Create( AProvider : TDataSetProvider)
16980: Create( ASocket : TCustomWinSocket; TimeOut : Longint)
16981: Create( ASocket : TSocket)
16982: Create( AStrings : TWideStrings)
16983: Create( AToolBar : TToolBar)
16984: Create( ATreeNodes : TTreeNodes)
16985: Create( Autofill : boolean)
16986: Create( AWebPageInfo : TAbstractWebPageInfo)
16987: Create( AWebRequest : TWebRequest)
16988: Create( Collection : TCollection)
16989: Create( Collection : TIdMessageParts; ABody : TStrings)
16990: Create( Collection : TIdMessageParts; const AFileName : TFileName)
16991: Create( Column : TColumn)
16992: Create( const AConvFamily : TConvFamily; const ADescription : string)
16993: Create( const AConvFamily : TConvFamily; const ADescription : string; const AFactor : Double)
16994: Create( const AConvFamily : TConvFamily; const ADescription : string; const AToCommonProc,
  AFromCommonProc : TConversionProc)
16995: Create( const AInitialState : Boolean; const AManualReset : Boolean)
16996: Create( const ATabSet : TTabSet)
16997: Create( const Compensate : Boolean)
16998: Create( const FileMap : TJclCustomFileMapping; Access, Size : Cardinal; ViewOffset : Int64)
16999: Create( const FileName : string)
17000: Create( const FileName : string; FileMode:Cardinal; const Name:string; Protect:Cardinal; const MaximumSize
  : Int64; const SecAttr : PSecurityAttributes);
17001: Create( const FileName : string; FileMode : WordfShareDenyWrite)
17002: Create( const MaskValue : string)
17003: Create(const Name:string; Protect:Cardinal;const MaximumSize:Int64;const SecAttr:PSecurityAttributes)
17004: Create( const Prefix : string)
17005: Create( const sRegularExpression : string; xFlags : TniRegularExpressionMatchFlags)
17006: Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
17007: Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
17008: Create( CoolBar : TCoolBar)
17009: Create( CreateSuspended : Boolean; ASocket : TServerClientWinSocket)
17010: Create( CreateSuspended : Boolean; ASocket : TServerWinSocket)
17011: Create( DataSet : TDataSet; const
  Text:Widestring;Options:TFilterOptions;ParserOptions:TParserOptions;const FieldName : Widestring;
  DepFields : TBits; FieldMap : TFieldMap)
17012: Create( DBCtrlGrid : TDBCtrlGrid)
17013: Create( DSTableProducer : TDSTableProducer)
17014: Create( DSTableProducer : TDSTableProducer; ColumnClass : THTMLTableColumnClass)
17015: Create( ErrorCode : DBIResult)
17016: Create( Field : TBlobField; Mode : TBlobStreamMode)
17017: Create( Grid : TCustomDBGrid; ColumnClass : TColumnClass)
17018: Create( HeaderControl : TCustomHeaderControl)
17019: Create( HTTPRequest : TWebRequest)
17020: Create( iStart : integer; sText : string)
17021: Create( iValue : Integer)
17022: Create( Kind : TMmTimerKind; Notification : TMmNotificationKind)
17023: Create( MciErrNo : MCIERROR; const Msg : string)
17024: Create(MemoryStream:TCustomMemStream;FreeStream:Boolean;const AIndexOption:TJclMappedTextReaderIndex);
17025: Create( Message : string; ErrorCode : DBResult)
17026: Create( Msg : string)
17027: Create( NativeError, Context : string; ErrCode, PrevError : Integer; E : Exception)
17028: Create( NativeError, Context : string; ErrorCode, PreviousError : DBResult)
17029: Create( oExpression : TniRegularExpression; eType : TniRegularExpressionStateType)
17030: Create( oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags)
17031: Create(oSource:TniRegularExpressionState;oDestination:TniRegularExpressionState;xCharacts:TCharSet;bLambda:bool)
17032: Create( Owner : EDBEngineError; ErrorCode : DBIResult; NativeError : Longint; Message : PChar)
17033: Create( Owner : TCustomComboBoxEx)
17034: CREATE( OWNER : TINDEXTDEFS; const NAME, FIELDS : string; OPTIONS : TINDEXTDEFS)
17035: Create( Owner : TPersistent)
17036: Create( Params : TStrings)
17037: Create( Size : Cardinal)
17038: Create( Socket : TSocket; ServerWinSocket : TServerWinSocket)
17039: Create( StatusBar : TCustomStatusBar)
17040: Create( WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass)
17041: Create( WebResponse : TWebResponse; ItemClass : TCollectionItemClass)
17042: Create(AHandle:Integer)
17043: Create(AOwner: TComponent); virtual;
17044: Create(const AURI : string)
17045: Create(FileName:String;Mode:Word)
17046: Create(Instance:THandle;ResName:String;ResType:PChar)
17047: Create(Stream : TStream)

```

```

17048: Create1( ADataset : TDataset);
17049: Create1(const FileHandle:THandle;const Name:string;Protect:Cardinal;const MaximumSize:Integer;const
SecAttr:PSecurityAttributes);
17050: Create1( const FileName : string; const AIndexOption : TJclMappedTextReaderIndex);
17051: Create2( Other : TObject);
17052: CreateAt(FileMap: TJclCustomFileMapping; Access,Size:Cardinal;ViewOffset:Integer;Address: Pointer)
17053: CreateError(const anErrCode: Integer;const asReplyMessage: string; const asErrorMessage: string)
17054: CreateFmt( MciErrNo : MCIERROR; const Msg : string; const Args : array of const)
17055: CreateFromId(Instance:THandle;ResId:Integer;ResType:PChar)
17056: CreateLinked( DBCtrlGrid : TDBCtrlGrid)
17057: CREATENEW(AOWNER:TComponent; Dummy: Integer)
17058: CreateRes( Ident : Integer);
17059: CreateRes( MciErrNo : MCIERROR; Ident : Integer)
17060: CreateRes( ResStringRec : PResStringRec);
17061: CreateResHelp( Ident : Integer; AHelpContext : Integer);
17062: CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Integer);
17063: CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
17064: CreateSize( AWidth, AHeight : Integer)
17065: Open( const Name : string; const InheritHandle : Boolean; const DesiredAccess : Cardinal)
17066:
17067: -----
17068: unit uPSI_MathMax;
17069: -----
17070: CONSTS
17071: Bernstein: Float = 0.2801694990238691330364364912307; // Bernstein constant
17072: Cbrt2: Float = 1.2599210498948731647672106072782; // CubeRoot(2)
17073: Cbrt3: Float = 1.4422495703074083823216383107801; // CubeRoot(3)
17074: Cbrt10: Float = 2.1544346900318837217592935665194; // CubeRoot(10)
17075: Cbrt100: Float = 4.6415888336127788924100763509194; // CubeRoot(100)
17076: CbrtPi: Float = 1.4645918875615232630201425272638; // CubeRoot(Pi)
17077: Catalan: Float = 0.9159655941772190150546035149324; // Catalan constant
17078: Pi: Float = 3.1415926535897932384626433832795; // PI
17079: PI: Extended = 3.1415926535897932384626433832795;
17080: PiOn2: Float = 1.5707963267948966192313216916398; // PI / 2
17081: PiOn3: Float = 1.0471975511965977461542144610932; // PI / 3
17082: PiOn4: Float = 0.78539816339744830961566084581988; // PI / 4
17083: Sqrt2: Float = 1.4142135623730950488016887242097; // Sqrt(2)
17084: Sqrt3: Float = 1.7320508075688772935274463415059; // Sqrt(3)
17085: Sqrt5: Float = 2.2360679774997896964091736687313; // Sqrt(5)
17086: Sqrt10: Float = 3.1622776601683793319988935444327; // Sqrt(10)
17087: SqrtPi: Float = 1.7724538509055160272981674833411; // Sqrt(Pi)
17088: Sqrt2Pi: Float = 2.506628274631000502415765284811; // Sqrt(2 * PI)
17089: TwoPi: Float = 6.283185307179586476925286766559; // 2 * PI
17090: ThreePi: Float = 9.4247779607693797153879301498385; // 3 * PI
17091: Ln2: Float = 0.69314718055994530941723212145818; // Ln(2)
17092: Ln10: Float = 2.3025850929940456840179914546844; // Ln(10)
17093: LnPi: Float = 1.1447298858494001741434273513531; // Ln(Pi)
17094: Log2J: Float = 0.30102999566398119521373889472449; // Log10(2)
17095: Log3: Float = 0.47712125471966243729502790325512; // Log10(3)
17096: LogPi: Float = 0.4971498726941338543512682882909; // Log10(Pi)
17097: LogE: Float = 0.43429448190325182765112891891661; // Log10(E)
17098: E: Float = 2.7182818284590452353602874713527; // Natural constant
17099: hLn2Pi: Float = 0.91893853320467274178032973640562; // Ln(2*PI)/2
17100: inv2Pi: Float = 0.15915494309189533576888376337251436203445964574046; // 0.5/Pi
17101: TwoToPower63: Float = 9223372036854775808.0; // 2^63
17102: GoldenMean: Float = 1.618033988749894848204586834365638; // GoldenMean
17103: EulerMascheroni: Float = 0.5772156649015328606065120900824; // Euler GAMMA
17104: RadCor : Double = 57.29577951308232; {number of degrees in a radian}
17105: StDelta : Extended = 0.00001; {delta for difference equations}
17106: StEpsilon : Extended = 0.00001; {epsilon for difference equations}
17107: StMaxIterations : Integer = 100; {max attempts for convergence}
17108:
17109: procedure SIRegister_StdConvs(CL: TPSPascalCompiler);
17110: begin
17111: MetersPerInch = 0.0254; // [1]
17112: MetersPerFoot = MetersPerInch * 12;
17113: MetersPerYard = MetersPerFoot * 3;
17114: MetersPerMile = MetersPerFoot * 5280;
17115: MetersPerNauticalMiles = 1852;
17116: MetersPerAstronomicalUnit = 1.49598E11; // [4]
17117: MetersPerLightSecond = 2.99792458E8; // [5]
17118: MetersPerLightYear = MetersPerLightSecond * 31556925.9747; // [7]
17119: MetersPerParsec = MetersPerAstronomicalUnit * 206264.806247096; // 60 * 60 * (180 / Pi)
17120: MetersPerCubit = 0.4572; // [6][7]
17121: MetersPerFathom = MetersPerFoot * 6;
17122: MetersPerFurlong = MetersPerYard * 220;
17123: MetersPerHand = MetersPerInch * 4;
17124: MetersPerPace = MetersPerInch * 30;
17125: MetersPerRod = MetersPerFoot * 16.5;
17126: MetersPerChain = MetersPerRod * 4;
17127: MetersPerLink = MetersPerChain / 100;
17128: MetersPerPoint = MetersPerInch * 0.013837; // [7]
17129: MetersPerPica = MetersPerPoint * 12;
17130:
17131: SquareMetersPerSquareInch = MetersPerInch * MetersPerInch;
17132: SquareMetersPerSquareFoot = MetersPerFoot * MetersPerFoot;
17133: SquareMetersPerSquareYard = MetersPerYard * MetersPerYard;
17134: SquareMetersPerSquareMile = MetersPerMile * MetersPerMile;
17135: SquareMetersPerAcre = SquareMetersPerSquareYard * 4840;

```

```

17136: SquareMetersPerSquareRod = MetersPerRod * MetersPerRod;
17137:
17138: CubicMetersPerCubicInch = MetersPerInch * MetersPerInch * MetersPerInch;
17139: CubicMetersPerCubicFoot = MetersPerFoot * MetersPerFoot * MetersPerFoot;
17140: CubicMetersPerCubicYard = MetersPerYard * MetersPerYard * MetersPerYard;
17141: CubicMetersPerCubicMile = MetersPerMile * MetersPerMile * MetersPerMile;
17142: CubicMetersPerAcreFoot = SquareMetersPerAcre * MetersPerFoot;
17143: CubicMetersPerAcreInch = SquareMetersPerAcre * MetersPerInch;
17144: CubicMetersPerCord = CubicMetersPerCubicFoot * 128;
17145: CubicMetersPerCordFoot = CubicMetersPerCubicFoot * 16;
17146:
17147: CubicMetersPerUSFluidGallon = CubicMetersPerCubicInch * 231; // [2][3][7]
17148: CubicMetersPerUSFluidQuart = CubicMetersPerUSFluidGallon / 4;
17149: CubicMetersPerUSFluidPint = CubicMetersPerUSFluidQuart / 2;
17150: CubicMetersPerUSFluidCup = CubicMetersPerUSFluidPint / 2;
17151: CubicMetersPerUSFluidGill = CubicMetersPerUSFluidCup / 2;
17152: CubicMetersPerUSFluidOunce = CubicMetersPerUSFluidCup / 8;
17153: CubicMetersPerUSFluidTablespoon = CubicMetersPerUSFluidOunce / 2;
17154: CubicMetersPerUSFluidTeaspoon = CubicMetersPerUSFluidOunce / 6;
17155:
17156: CubicMetersPerUSDryGallon = CubicMetersPerCubicInch * 268.8025; // [7]
17157: CubicMetersPerUSDryQuart = CubicMetersPerUSDryGallon / 4;
17158: CubicMetersPerUSDryPint = CubicMetersPerUSDryQuart / 2;
17159: CubicMetersPerUSDryPeck = CubicMetersPerUSDryGallon * 2;
17160: CubicMetersPerUSDryBucket = CubicMetersPerUSDryPeck * 2;
17161: CubicMetersPerUSDryBushel = CubicMetersPerUSDryBucket * 2;
17162:
17163: CubicMetersPerUKGallon = 0.00454609; // [2][7]
17164: CubicMetersPerUKPottle = CubicMetersPerUKGallon / 2;
17165: CubicMetersPerUKQuart = CubicMetersPerUKPottle / 2;
17166: CubicMetersPerUKPint = CubicMetersPerUKQuart / 2;
17167: CubicMetersPerUKGill = CubicMetersPerUKPint / 4;
17168: CubicMetersPerUKOunce = CubicMetersPerUKPint / 20;
17169: CubicMetersPerUKPeck = CubicMetersPerUKGallon * 2;
17170: CubicMetersPerUKBucket = CubicMetersPerUKPeck * 2;
17171: CubicMetersPerUKBushel = CubicMetersPerUKBucket * 2;
17172:
17173: GramsPerPound = 453.59237; // [1][7]
17174: GramsPerDrams = GramsPerPound / 256;
17175: GramsPerGrains = GramsPerPound / 7000;
17176: GramsPerTons = GramsPerPound * 2000;
17177: GramsPerLongTons = GramsPerPound * 2240;
17178: GramsPerOunces = GramsPerPound / 16;
17179: GramsPerStones = GramsPerPound * 14;
17180:
17181: MaxAngle', ( 9223372036854775808.0);
17182: MaxTanH', ( 5678.2617031470719747459655389854);
17183: MaxFactorial', 'LongInt').SetInt( 1754);
17184: MaxFloatingPoint', (1.189731495357231765085759326628E+4932);
17185: MinFloatingPoint', (3.3621031431120935062626778173218E-4932);
17186: MaxTanH', ( 354.89135644669199842162284618659);
17187: MaxFactorial', 'LongInt').SetInt( 170);
17188: MaxFloatingPointD', (1.797693134862315907729305190789E+308);
17189: MinFloatingPointD', (2.2250738585072013830902327173324E-308);
17190: MaxTanH', ( 44.361419555836499802702855773323);
17191: MaxFactorial', 'LongInt').SetInt( 33);
17192: MaxFloatingPointS', ( 3.4028236692093846346337460743177E+38);
17193: MinFloatingPointS', ( 1.1754943508222875079687365372222E-38);
17194: PiExt', ( 3.1415926535897932384626433832795);
17195: RatioDegToRad', ( PiExt / 180.0);
17196: RatioGradToRad', ( PiExt / 200.0);
17197: RatioDegToGrad', ( 200.0 / 180.0);
17198: RatioGradToDeg', ( 180.0 / 200.0);
17199: Crc16PolynomCCITT', 'LongWord').SetUInt( $1021);
17200: Crc16PolynomIBM', 'LongWord').SetUInt( $8005);
17201: Crc16Bits', 'LongInt').SetInt( 16);
17202: Crc16Bytes', 'LongInt').SetInt( 2);
17203: Crc16HighBit', 'LongWord').SetUInt( $8000);
17204: NotCrc16HighBit', 'LongWord').SetUInt( $7FFF);
17205: Crc32PolynomIEEE', 'LongWord').SetUInt( $04C11DB7);
17206: Crc32PolynomCastagnoli', 'LongWord').SetUInt( $1EDC6F41);
17207: Crc32Koopman', 'LongWord').SetUInt( $741B8CD7);
17208: Crc32Bits', 'LongInt').SetInt( 32);
17209: Crc32Bytes', 'LongInt').SetInt( 4);
17210: Crc32HighBit', 'LongWord').SetUInt( $80000000);
17211: NotCrc32HighBit', 'LongWord').SetUInt( $7FFFFFFF);
17212:
17213: MinByte = Low(Byte);
17214: MaxByte = High(Byte);
17215: MinWord = Low(Word);
17216: MaxWord = High(Word);
17217: MinShortInt = Low(ShortInt);
17218: MaxShortInt = High(ShortInt);
17219: MinSmallInt = Low(SmallInt);
17220: MaxSmallInt = High(SmallInt);
17221: MinLongWord = LongWord(Low(LongWord));
17222: MaxLongWord = LongWord(High(LongWord));
17223: MinLongInt = LongInt(Low(LongInt));
17224: MaxLongInt = LongInt(High(LongInt));

```



```

17225: MinInt64      = Int64(Low(Int64));
17226: MaxInt64      = Int64(High(Int64));
17227: MinInteger    = Integer(Low(Integer));
17228: MaxInteger    = Integer(High(Integer));
17229: MinCardinal   = Cardinal(Low(Cardinal));
17230: MaxCardinal   = Cardinal(High(Cardinal));
17231: MinNativeUInt = NativeUInt(Low(NativeUInt));
17232: MaxNativeUInt = NativeUInt(High(NativeUInt));
17233: MinNativeInt  = NativeInt(Low(NativeInt));
17234: MaxNativeInt  = NativeInt(High(NativeInt));
17235: Function CosH( const Z : Float) : Float;
17236: Function SinH( const Z : Float) : Float;
17237: Function TanH( const Z : Float) : Float;
17238:
17239:
17240: //*****from DMath.Dll Lib of types.inc in source\dmath_dll
17241: InvLn2      = 1.44269504088896340736; { 1/Ln(2) }
17242: InvLn10     = 0.43429448190325182765; { 1/Ln(10) }
17243: TwoPi       = 6.28318530717958647693; { 2*Pi }
17244: PiDiv2      = 1.57079632679489661923; { Pi/2 }
17245: SqrtPi      = 1.77245385090551602730; { Sqrt(Pi) }
17246: Sqrt2Pi     = 2.50662827463100050242; { Sqrt(2*Pi) }
17247: InvSqrt2Pi  = 0.39894228040143267794; { 1/Sqrt(2*Pi) }
17248: LnSqrt2Pi   = 0.91893853320467274178; { Ln(Sqrt(2*Pi)) }
17249: Ln2PiDiv2   = 0.91893853320467274178; { Ln(2*Pi)/2 }
17250: Sqrt2       = 1.41421356237309504880; { Sqrt(2) }
17251: Sqrt2Div2   = 0.70710678118654752440; { Sqrt(2)/2 }
17252: Gold        = 1.61803398874989484821; { Golden Mean = (1 + Sqrt(5))/2 }
17253: CGold       = 0.38196601125010515179; { 2 - GOLD }
17254: MachEp      = 2.220446049250313E-16; { 2^(-52) }
17255: MaxNum      = 1.797693134862315E+308; { 2^1024 }
17256: MinNum      = 2.225073858507202E-308; { 2^(-1022) }
17257: MaxLog      = 709.7827128933840;
17258: MinLog      = -708.3964185322641;
17259: MaxFac      = 170;
17260: MaxGam      = 171.624376956302;
17261: MaxLgm      = 2.556348E+305;
17262: SingleCompareDelta = 1.0E-34;
17263: DoubleCompareDelta = 1.0E-280;
17264: { $IFDEF CLR }
17265: ExtendedCompareDelta = DoubleCompareDelta;
17266: { $ELSE }
17267: ExtendedCompareDelta = 1.0E-4400;
17268: { $ENDIF }
17269: Bytes1KB    = 1024;
17270: Bytes1MB    = 1024 * Bytes1KB;
17271: Bytes1GB    = 1024 * Bytes1MB;
17272: Bytes64KB   = 64 * Bytes1KB;
17273: Bytes64MB   = 64 * Bytes1MB;
17274: Bytes2GB    = 2 * LongWord(Bytes1GB);
17275: clBlack32'  = $FF000000 );
17276: clDimGray32' = $FF3F3F3F );
17277: clGray32'   = $FF7F7F7F );
17278: clLightGray32' = $FFBFBFBF );
17279: clWhite32'  = $FFFFFFF );
17280: clMaroon32' = $FF7F0000 );
17281: clGreen32'  = $FF007F00 );
17282: clOlive32'  = $FF7F7F00 );
17283: clNavy32'   = $FF00007F );
17284: clPurple32' = $FF7F007F );
17285: clTeal32'   = $FF007F7F );
17286: clRed32'    = $FFFF0000 );
17287: clLime32'   = $FF00FF00 );
17288: clYellow32' = $FFFFFFF0 );
17289: clBlue32'   = $FF0000FF );
17290: clFuchsia32' = $FFFF00FF );
17291: clAqua32'   = $FF00FFFF );
17292: clAliceBlue32' = $FFF0F8FF );
17293: clAntiqueWhite32' = $FFFAEBD7 );
17294: clAquamarine32' = $FFF7FFD4 );
17295: clAzure32'   = $FFF0FFFF );
17296: clBeige32'   = $FFF5F5DC );
17297: clBisque32'  = $FFFFE4C4 );
17298: clBlancheDalmond32' = $FFFFEBCD );
17299: clBlueViolet32' = $FF8A2BE2 );
17300: clBrown32'   = $FFA52A2A );
17301: clBurlyWood32' = $FFDEB887 );
17302: clCadetBlue32' = $FF5F9EAO );
17303: clChartReuse32' = $FF7FFF00 );
17304: clChocolate32' = $FFD2691E );
17305: clCoral32'   = $FFF7F50 );
17306: clCornFlowerBlue32' = $FF6495ED );
17307: clCornSilk32' = $FFFFFF8DC );
17308: clCrimson32' = $FFDC143C );
17309: clDarkBlue32' = $FF00008B );
17310: clDarkCyan32' = $FF008B8B );
17311: clDarkGoldenRod32' = $FFB8860B );
17312: clDarkGray32' = $FFA9A9A9 );
17313: clDarkGreen32' = $FF006400 );

```

```
17314: clDarkGrey32', $FFA9A9A9 );
17315: clDarkKhaki32', $FFBDB76B );
17316: clDarkMagenta32', $FF8B008B );
17317: clDarkOliveGreen32', $FF556B2F );
17318: clDarkOrange32', $FFFF8C00 );
17319: clDarkOrchid32', $FF9932CC );
17320: clDarkRed32', $FF8B0000 );
17321: clDarkSalmon32', $FFE9967A );
17322: clDarkSeaGreen32', $FF8FBC8F );
17323: clDarkSlateBlue32', $FF483D8B );
17324: clDarkSlateGray32', $FF2F4F4F );
17325: clDarkSlateGrey32', $FF2F4F4F );
17326: clDarkTurquoise32', $FF00CED1 );
17327: clDarkViolet32', $FF9400D3 );
17328: clDeepPink32', $FFFFFF1493 );
17329: clDeepSkyBlue32', $FF00BFFF );
17330: clDodgerBlue32', $FF1E90FF );
17331: clFireBrick32', $FFB22222 );
17332: clFloralWhite32', $FFFFFFFA0 );
17333: clGainsboro32', $FFDCDCDC );
17334: clGhostWhite32', $FFF8F8FF );
17335: clGold32', $FFFD700 );
17336: clGoldenRod32', $FFDAA520 );
17337: clGreenYellow32', $FFADFF2F );
17338: clGrey32', $FF808080 );
17339: clHoneyDew32', $FFF0FFF0 );
17340: clHotPink32', $FFFF69B4 );
17341: clIndianRed32', $FFCD5C5C );
17342: clIndigo32', $FF4B0082 );
17343: clIvory32', $FFFFFFF0 );
17344: clKhaki32', $FFF0E68C );
17345: clLavender32', $FFE6E6FA );
17346: clLavenderBlush32', $FFFFFFF0F5 );
17347: clLawnGreen32', $FF7CFC00 );
17348: clLemonChiffon32', $FFFFFFFACD );
17349: clLightBlue32', $FFADD8E6 );
17350: clLightCoral32', $FFF08080 );
17351: clLightCyan32', $FFE0FFFF );
17352: clLightGoldenRodYellow32', $FFFAFAD2 );
17353: clLightGreen32', $FFF0EE90 );
17354: clLightGrey32', $FFD3D3D3 );
17355: clLightPink32', $FFFB6C1 );
17356: clLightSalmon32', $FFFA07A );
17357: clLightSeagreen32', $FF20B2AA );
17358: clLightSkyblue32', $FF87CEFA );
17359: clLightSlategray32', $FF778899 );
17360: clLightSlategrey32', $FF778899 );
17361: clLightSteelblue32', $FFB0C4DE );
17362: clLightYellow32', $FFFFFFF0 );
17363: clLtGray32', $FFC0C0C0 );
17364: clMedGray32', $FFA0A0A4 );
17365: clDkGray32', $FF808080 );
17366: clMoneyGreen32', $FFC0DCC0 );
17367: clLegacySkyBlue32', $FFA6CAF0 );
17368: clCream32', $FFFFFFBF0 );
17369: clLimeGreen32', $FF32CD32 );
17370: clLinen32', $FFFAF0E6 );
17371: clMediumAquamarine32', $FF66CDAA );
17372: clMediumBlue32', $FF0000CD );
17373: clMediumOrchid32', $FFBA55D3 );
17374: clMediumPurple32', $FF9370DB );
17375: clMediumSeaGreen32', $FF3CB371 );
17376: clMediumSlateBlue32', $FF7B68EE );
17377: clMediumSpringGreen32', $FF00FA9A );
17378: clMediumTurquoise32', $FF48D1CC );
17379: clMediumVioletRed32', $FFC71585 );
17380: clMidnightBlue32', $FF191970 );
17381: clMintCream32', $FFF5FFFA );
17382: clMistyRose32', $FFFFE4E1 );
17383: clMoccasin32', $FFFE4B5 );
17384: clNavajoWhite32', $FFFFDEAD );
17385: clOldLace32', $FFFD5E6 );
17386: clOliveDrab32', $FF6B8E23 );
17387: clOrange32', $FFFA500 );
17388: clOrangeRed32', $FFFF4500 );
17389: clOrchid32', $FFDA70D6 );
17390: clPaleGoldenRod32', $FFEE8AA );
17391: clPaleGreen32', $FF98FB98 );
17392: clPaleTurquoise32', $FFAFEEEE );
17393: clPaleVioletred32', $FFDB7093 );
17394: clPapayaWhip32', $FFFFEFD5 );
17395: clPeachPuff32', $FFFFDAB9 );
17396: clPeru32', $FFCD853F );
17397: clPlum32', $FFDDA0DD );
17398: clPowderBlue32', $FFB0E0E6 );
17399: clRosyBrown32', $FFBC8F8F );
17400: clRoyalBlue32', $FF4169E1 );
17401: clSaddleBrown32', $FF8B4513 );
17402: clSalmon32', $FFFA8072 );
```

```

17403:   clSandyBrown32', $FFF4A460 ));
17404:   clSeaGreen32', $FF2E8B57 ));
17405:   clSeaShell32', $FFFFFF5EE ));
17406:   clSienna32', $FFA0522D ));
17407:   clSilver32', $FFC0C0C0 ));
17408:   clSkyblue32', $FF87CEEB ));
17409:   clSlateBlue32', $FF6A5ACD ));
17410:   clSlateGray32', $FF708090 ));
17411:   clSlateGrey32', $FF708090 ));
17412:   clSnow32', $FFFFFFAFA ));
17413:   clSpringgreen32', $FF00FF7F ));
17414:   clSteelblue32', $FF4682B4 ));
17415:   clTan32', $FFD2B48C ));
17416:   clThistle32', $FFD8BFD8 ));
17417:   clTomato32', $FFFF6347 ));
17418:   clTurquoise32', $FF40E0D0 ));
17419:   clViolet32', $FFEE82EE ));
17420:   clWheat32', $FFF5DEB3 ));
17421:   clWhitesmoke32', $FFF5F5F5 ));
17422:   clYellowgreen32', $FF9ACD32 ));
17423:   clTrWhite32', $7FFFFFFF ));
17424:   clTrBlack32', $7F000000 ));
17425:   clTrRed32', $7FFF0000 ));
17426:   clTrGreen32', $7F00FF00 ));
17427:   clTrBlue32', $7F0000FF ));
17428:   // Fixed point math constants
17429:   FixedOne = $10000; FixedHalf = $7FFF;
17430:   FixedPI = Round(PI * FixedOne);
17431:   FixedToFloat = 1/FixedOne;
17432:
17433:   Special Types
17434:   *****
17435:   type Complex = record
17436:     X, Y : Float;
17437:   end;
17438:   type TVector = array of Float;
17439:   TIntVector = array of Integer;
17440:   TCompVector = array of Complex;
17441:   TBoolVector = array of Boolean;
17442:   TStrVector = array of String;
17443:   TMatrix = array of TVector;
17444:   TIntMatrix = array of TIntVector;
17445:   TCompMatrix = array of TCompVector;
17446:   TBoolMatrix = array of TBoolVector;
17447:   TStrMatrix = array of TStrVector;
17448:   TByteArray = array[0..32767] of byte; !
17449:   THexArray = array[0..15] of Char; // = '0123456789ABCDEF';
17450:   TBitmapStyle = (bsNormal, bsCentered, bsStretched);
17451:   T2StringArray = array of array of string;
17452:   T2IntegerArray = array of array of integer;
17453:   AddTypeS('INT_PTR', 'Integer
17454:   AddTypeS('LONG_PTR', 'Integer
17455:   AddTypeS('UINT_PTR', 'Cardinal
17456:   AddTypeS('ULONG_PTR', 'Cardinal
17457:   AddTypeS('DWORD_PTR', 'ULONG_PTR
17458:   TIntegerDynArray', 'array of Integer
17459:   TCardinalDynArray', 'array of Cardinal
17460:   TWordDynArray', 'array of Word
17461:   TSmallIntDynArray', 'array of SmallInt
17462:   TByteDynArray', 'array of Byte
17463:   TShortIntDynArray', 'array of ShortInt
17464:   TInt64DynArray', 'array of Int64
17465:   TLongWordDynArray', 'array of LongWord
17466:   TSingleDynArray', 'array of Single
17467:   TDoubleDynArray', 'array of Double
17468:   TBooleanDynArray', 'array of Boolean
17469:   TStringDynArray', 'array of string
17470:   TWideStringDynArray', 'array of WideString
17471:   TDynByteArray = array of Byte;
17472:   TDynShortintArray = array of Shortint;
17473:   TDynSmallintArray = array of Smallint;
17474:   TDynWordArray = array of Word;
17475:   TDynIntegerArray = array of Integer;
17476:   TDynLongintArray = array of Longint;
17477:   TDynCardinalArray = array of Cardinal;
17478:   TDynInt64Array = array of Int64;
17479:   TDynExtendedArray = array of Extended;
17480:   TDynDoubleArray = array of Double;
17481:   TDynSingleArray = array of Single;
17482:   TDynFloatArray = array of Float;
17483:   TDynPointerArray = array of Pointer;
17484:   TDynStringArray = array of string;
17485:   TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
17486:     ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
17487:   TSynSearchOptions = set of TSynSearchOption;
17488:
17489:
17490:
17491: /* Project : Base Include RunTime Lib for maXbox *Name: pas_includebox.inc

```

```

17492: -----
17493: procedure drawPolygon(vPoints: TXYVector; cFrm: TForm);
17494: procedure drawPlot(vPoints: TXYVector; cFrm: TForm; vcolor: integer);
17495: procedure SaveCanvas(vCanvas: TCanvas; FileName: string);
17496: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
17497: function CheckStringSum(vstring: string): integer;
17498: function HexToInt(HexNum: string): LongInt;
17499: function IntToBin(Int: Integer): String;
17500: function BinToInt(Binary: String): Integer;
17501: function HexToBin(HexNum: string): string; external2
17502: function BinToHex(Binary: String): string;
17503: function IntToFloat(i: Integer): double;
17504: function AddThousandSeparator(S: string; myChr: Char): string;
17505: function Max3(const X,Y,Z: Integer): Integer;
17506: procedure Swap(var X,Y: char); // faster without inline
17507: procedure ReverseString(var S: String);
17508: function CharToHexStr(Value: Char): string;
17509: function CharToUnicode(Value: Char): string;
17510: function Hex2Dec(Value: Str002): Byte;
17511: function HexStrCodeToStr(Value: string): string;
17512: function HexToStr(i: integer; value: string): string;
17513: function UnicodeToStr(Value: string): string;
17514: function CRC16(statement: string): string;
17515: function SearchForSubstrings(aStrList: TStrings; aSearchStr1, aSearchStr2: string): string;
17516: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string);
17517: procedure ShowInterfaces(myFile: string);
17518: function Fact2(av: integer): extended;
17519: function BoolToStr(B: Boolean): string;
17520: function GCD(x, y : LongInt) : LongInt;
17521: function LCM(m,n: longint): longint;
17522: function GetASCII: string;
17523: function GetItemHeight(Font: TFont): Integer;
17524: function myPlaySound(s: pchar; flag, syncflag: integer): boolean;
17525: function myGetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
17526: function getHINSTANCE: longword;
17527: function getHMODULE: longword;
17528: function GetASCII: string;
17529: function ByteIsOk(const AByte: string; var VB: Byte): boolean;
17530: function WordIsOk(const AWord: string; var VW: Word): boolean;
17531: function TwentyFourBitValueIsOk(const AValue: string; var VI: Integer): boolean;
17532: function LongIsOk(const ALong: string; var VC: Cardinal): boolean;
17533: function SafeStr(const s: string): string;
17534: function ExtractUrlPath(const FileName: string): string;
17535: function ExtractUrlName(const FileName: string): string;
17536: function IsInternet: boolean;
17537: function RotateLeft1Bit_u32( Value: uint32): uint32;
17538: procedure LinearRegression(const KnownY:array of Double;const KnownX: array of Double;NData:Int;var
LF:TStLinEst; ErrorStats : Boolean);
17539: procedure getEnvironmentInfo;
17540: procedure AntiFreeze;
17541: function GetCPUSpeed: Double;
17542: function IsVirtualPcGuest : Boolean;
17543: function IsVmWareGuest : Boolean;
17544: procedure StartSerialDialog;
17545: function IsWoW64: boolean;
17546: function IsWow64String(var s: string): Boolean;
17547: procedure StartThreadDemo;
17548: function RGB(R,G,B: Byte): TColor;
17549: function SendLn(amess: string): boolean;
17550: procedure maXbox;
17551: function AspectRatio(aWidth, aHeight: Integer): String;
17552: function wget(aURL, afile: string): boolean;
17553: procedure PrintList(Value: TStringList);
17554: procedure PrintImage(aValue: TBitmap; Style: TBitmapStyle);
17555: procedure getEnvironmentInfo;
17556: procedure AntiFreeze;
17557: function getBitmap(apath: string): TBitmap;
17558: procedure ShowMessageBig(const aText : string);
17559: function YesNoDialog(const ACaption, AMsg: string): boolean;
17560: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);
17561: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
17562: //function myStrToBytes(const Value: String): TBytes;
17563: //function myBytesToStr(const Value: TBytes): String;
17564: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
17565: function getBitmap(apath: string): TBitmap;
17566: procedure ShowMessageBig(const aText : string);
17567: function StrToBytes(const Value: String): TBytes;
17568: function BytesToStr(const Value: TBytes): String;
17569: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
17570: function ReverseDNSLookup(const IPAdrs:String;const DNSServer:String;Timeout,Retries:Int;var
HostName:String):Bool;
17571: function FindInPaths(const fileName, paths : String) : String;
17572: procedure initHexArray(var hexn: THexArray);
17573: function josephusG(n,k: integer; var graphout: string): integer;
17574: function isPowerof2(num: int64): boolean;
17575: function powerOf2(exponent: integer): int64;
17576: function getBigPI: string;
17577: procedure MakeSound(Frequency{Hz}, Duration{mSec}: Integer; Volume: TVolumeLevel; savefilePath: string);
17578: function GetASCIILine: string;

```



```

17579: procedure MakeComplexSound(N:integer{stream # to use}; freqlist:TStrings; Duration{mSec}: Integer;
17580:                               pinknoise: boolean; shape: integer; Volume: TVolumeLevel);
17581: procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
17582: procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
17583: function mapfunc(ax, in_min, in_max, out_min, out_max: integer): integer;
17584: function mapmax(ax, in_min, in_max, out_min, out_max: integer): integer;
17585: function isKeyPressed: boolean;
17586: function KeyPress: boolean;
17587: procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
17588: function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
17589: function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
17590: function GetOSName: string;
17591: function GetOSVersion: string;
17592: function GetOSNumber: string;
17593: function getEnvironmentString: string;
17594: procedure StrReplace(var Str: String; Old, New: String);
17595: procedure SendEmail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
17596: function getTeamViewerID: string;
17597: procedure RecurseDirectory(Dir:String; IncludeSubs:boolean; callback:TFileCallbackProcedure);
17598: procedure RecurseDirectory2(Dir : String; IncludeSubs : boolean);
17599: procedure WinInet_HttpGet(const Url: string; Stream:TStream);
17600: procedure GetQrCode2(Width,Height: Word;Correct_Level:string;const Data:string;apath:string);
17601: function StartSocketService: Boolean;
17602: procedure StartSocketServiceForm;
17603: function GetFileList(FileList: TStringlist; apath: string): TStringlist;
17604: function GetFileList1(apath: string): TStringlist;
17605: procedure LetFileList(FileList: TStringlist; apath: string);
17606: procedure StartWeb(aurl: string);
17607: function GetTodayFiles(startdir, amask: string): TStringlist;
17608: function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
17609: function JavahashCode(val: string): Integer;
17610: procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
17611: procedure SaveBytesToFile2(const Data: Sysutils.TBytes; const FileName: string);
17612: procedure HideWindowForSeconds(secs: integer); { //3 seconds}
17613: procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm); { //3 seconds}
17614: procedure ConvertToGray(Cnv: TCanvas);
17615: function GetFileDate(aFile:string; aWithTime:Boolean):string;
17616: procedure ShowMemory;
17617: function ShowMemory2: string;
17618: function getHostIP: string;
17619: procedure ShowBitmap(bmap: TBitmap);
17620: function GetOsVersionInfo: TOSVersionInfo; //thx to wischnewski
17621:
17622:
17623: // News of 3.9.8 up
17624: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
17625: Conversion Routines, Prebuild Forms, more RCDData, DebugOutString
17626: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
17627: JvChart - TjvChart Component - 2009 Public
17628: MemoryLeakReport in ini-file (MEMORYREPORT=Y)
17629: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
17630: TAdoQuery.SQL.Add() fixed, ShLwAPI extensions, Indy HTTPHeader Extensions
17631: DMATH DLL included incl. Demos
17632: Interface Navigator menu/View/Intf Navigator
17633: Unit Explorer menu/Debug/Units Explorer
17634: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maXcel
17635: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
17636: Script History to 9 Files WebServer light /Options/Addons/WebServer
17637: Full Text Finder, JVSIMLogic Simulator Package
17638: Halt-Stop Program in Menu, WebServer2, Stop Event ,
17639: Conversion Routines, Prebuild Forms, CodeSearch
17640: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
17641: Conversion Routines, Prebuild Forms, more RCDData, DebugOutString
17642: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
17643: JvChart - TjvChart Component - 2009 Public, mXGames, JvgXMLSerializer, TjvPaintFX
17644: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib
17645: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
17646: IDE Reflection API, Session Service Shell S3
17647: additional SynEdit API, isKeyPressed Routine, Bookmarks, OpenToolsAPI Catalog (OTAC)
17648: Class TMonitor, Configuration Tutorial maxbox_starter25.pdf, Chess.dll Game
17649: arduino map() function, PMRandom Generator
17650: StBarCode Lib, StreamReaderClass, BarCode Package, Astro Package
17651: more ShellAPI, add 32 more units, Simulated Annealing, GenAlgo
17652: REST Test Lib, Multilang Component, Forth Interpreter
17653: New Macros, Sendmail (instant email), DevCUnits, Tetris Addon
17654: DCOM, MDAC, MIDI, TLS support, Posmarks, Utils Addon
17655: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
17656: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
17657: First LCL of Lazarus, CmdLine API, ToDo List, 36 more Units preCompiled
17658: QRCode Service, add more CFunctions like CDateTime of Synapse
17659: Gamma Functions, IndyPackage4, HotLog Threadable, FormTemplateLibrary FTL
17660: Nonlinear regression, ADO Workbench Addon, Assign fixing, IntfNavigator fixing, Applet
17661: 30 more Units preCompiled, QRCode Indy Service, more CFunctions like CFill or SRand
17662: RestartDialog, RTF, SQL Scanner, RichEdit, 15 more Units
17663: Tool Section, SOAP Tester, Hot Log Logger2, TCPPortScan, 28 more Units
17664: BOLD Package, Indy Package5, maTRIX. MATHEMAX
17665: SPS Utils WDOS, Plc BitBus (PetriNet), 40 more units
17666: emacs layers: system-package-component-unit-class-function-block
17667: HighPrecision Timers, Indy Package6, AutoDetect, UltraForms

```

```

17668: Reversi, GOL, bugfixing, 8 more units, Tutorial 24 Clean Code
17669: Tutorial 18.3 RGB LED, OpenGL Geometry, maxpix, statictext
17670: OpenGL Game Demo: ..Options/Add Ons/Reversi
17671: IBUtils Refactor, InterBase Package, DotNet Routines (JvExControls)
17672: add 31 units, mX4 Introduction Paper, more Socket&Streams, ShortString Routines
17673: 7% performance gain (hot spot profiling)
17674: PEP -Pascal Education Program, GSM Module, CGI, PHP Runner
17675: add 42 + 22 (64 units), memcached database, autobookmark, Alcinoe PAC, IPC Lib
17676: Orpheus PAC, AsyncFree Library advapi32 samples, FirebirdExp+MySQL units
17677: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
17678:
17679: add routines in 3.9.7.5
17680: 097: procedure RIRegister_BarCodeScanner_Routines(S: TPSExec);
17681: 996: procedure RIRegister_DBCtrls_Routines(S: TPSExec);
17682: 069: procedure RIRegister_IdStrings_Routines(S: TPSExec);
17683: 516: procedure RIRegister_JclMultimedia_Routines(S: TPSExec);
17684: 215: procedure RIRegister_PNGLoader_Routines(S: TPSExec);
17685: 374: procedure RIRegister_SerDlgs_Routines(S: TPSExec);
17686: 777: procedure RIRegister_LinarBitmap_Routines(S: TPSExec);
17687:
17688: ////////////////////////////////// TestUnits //////////////////////////////////
17689: SelftestPEM;
17690: SelfTestCFundamentUtils;
17691: SelfTestCFileUtils;
17692: SelfTestCDateTime;
17693: SelfTestCTimer;
17694: SelfTestCRandom;
17695: Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter
17696: Assert(WinPathToUnixPath('c\d.f') = '/c/d.f', 'WinPathToUnixPath
17697:
17698: // Note: There's no need for installing a client certificate in the
17699: // webbrowser. The server asks the webbrowser to send a certificate but
17700: // if nothing is installed the software will work because the server
17701: // doesn't check to see if a client certificate was supplied. If you want you can install:
17702: //
17703: // file: c_cacert.p12
17704: // password: c_cakey
17705:
17706: TGraphicControl = class(TControl)
17707: private
17708: FCanvas: TCanvas;
17709: procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
17710: protected
17711: procedure Paint; virtual;
17712: property Canvas: TCanvas read FCanvas;
17713: public
17714: constructor Create(AOwner: TComponent); override;
17715: destructor Destroy; override;
17716: end;
17717:
17718: TCustomControl = class(TWinControl)
17719: private
17720: FCanvas: TCanvas;
17721: procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
17722: protected
17723: procedure Paint; virtual;
17724: procedure PaintWindow(DC: HDC); override;
17725: property Canvas: TCanvas read FCanvas;
17726: public
17727: constructor Create(AOwner: TComponent); override;
17728: destructor Destroy; override;
17729: end;
17730: RegisterPublishedProperties;
17731: ('ONCHANGE', 'TNotifyEvent', iptrw);
17732: ('ONCLICK', 'TNotifyEvent', iptrw);
17733: ('ONDBLCLICK', 'TNotifyEvent', iptrw);
17734: ('ONENTER', 'TNotifyEvent', iptrw);
17735: ('ONEXIT', 'TNotifyEvent', iptrw);
17736: ('ONKEYDOWN', 'TKeyEvent', iptrw);
17737: ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
17738: ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
17739: ('ONMOUSEMOVE', 'TMouseMoveEvent', iptrw);
17740: ('ONMOUSEUP', 'TMouseEvent', iptrw);
17741: //*****
17742: // To stop the while loop, click on Options/Show Include (boolean switch)!
17743: Control a loop in a script with a form event:
17744: IncludeON; //control the while loop
17745: while maxform1.ShowInclude1.checked do begin //menu event Options/Show Include
17746:
17747: //-----
17748: //*****mX4 ini-file Configuration*****
17749: //-----
17750: using config file maxboxdef.ini menu/Help/Config File
17751:
17752: /*** Definitions for maXbox mX3 ***/
17753: [FORM]
17754: LAST_FILE=E:\maXbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
17755: FONTSIZE=14
17756: EXTENSION=txt

```

```

17757: SCREENX=1386
17758: SCREENY=1077
17759: MEMHEIGHT=350
17760: PRINTFONT=Courier New //GUI Settings
17761: LINENUMBERS=Y //line numbers at gutter in editor at left side
17762: EXCEPTIONLOG=Y //store excepts and success in 2 log files see below! - menu Debug/Show Last Exceptions
17763: EXECUTESHELL=Y //prevents execution of ExecuteShell() or ExecuteCommand()
17764: BOOTSCRIPT=Y //enabling load a boot script
17765: MEMORYREPORT=Y //shows memory report on closing maXbox
17766: MACRO=Y //expand macros (see below) in code e.g. #path:E:\maxbox\maxbox3\docs\
17767: NAVIGATOR=N //shows function list at the right side of editor
17768: NAVWIDTH=350 //width of the right side interface list <CTRL L>
17769: AUTOBOOKMARK=Y //sets on all functions a bookmark to jump
17770: [WEB]
17771: IPPORT=8080 //for internal webserver - menu /Options/Add Ons/WebServer
17772: IPHOST=192.168.1.53
17773: ROOTCERT=filepathY
17774: SCERT=filepathY
17775: RSAKEY=filepathY
17776: VERSIONCHECK=Y
17777:
17778: using Logfile: maxboxlog.log , Exceptionlogfile: maxboxerrorlog.txt
17779:
17780: Also possible to set report memory in script to override ini setting
17781: procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
17782:
17783:
17784: After Change the ini file you can reload the file with ../Help/Config Update
17785:
17786: //-----
17787: //*****mX4 maildef.ini ini-file Configuration*****
17788: //-----
17789: /*** Definitions for maXMail ***
17790: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
17791: [MAXMAIL]
17792: HOST=getmail.softwareschule.ch
17793: USER=mailusername
17794: PASS=password
17795: PORT=110
17796: SSL=Y
17797: BODY=Y
17798: LAST=5
17799:
17800: ADO Connection String:
17801: Provider=MSDASQL;DSN=mx3base;Uid=sa;Pwd=admin
17802:
17803: OpenSSL Lib: unit ssl_openssl_lib;
17804: {$IFDEF CIL}
17805: const
17806: {$IFDEF LINUX}
17807: DLLSSLName = 'libssl.so';
17808: DLLUtilName = 'libcrypto.so';
17809: {$ELSE}
17810: DLLSSLName = 'ssleay32.dll';
17811: DLLUtilName = 'libeay32.dll';
17812: {$ENDIF}
17813: {$ELSE}
17814: var
17815: {$IFDEF MSWINDOWS}
17816: {$IFDEF DARWIN}
17817: DLLSSLName: string = 'libssl.dylib';
17818: DLLUtilName: string = 'libcrypto.dylib';
17819: {$ELSE}
17820: DLLSSLName: string = 'libssl.so';
17821: DLLUtilName: string = 'libcrypto.so';
17822: {$ENDIF}
17823: {$ELSE}
17824: DLLSSLName: string = 'ssleay32.dll';
17825: DLLSSLName2: string = 'libssl32.dll';
17826: DLLUtilName: string = 'libeay32.dll';
17827: {$ENDIF}
17828: {$ENDIF}
17829:
17830:
17831:
17832: //-----
17833: //*****mX4 Macro Tags *****
17834: //-----
17835:
17836: asm #name #hostmAPSN2APSN2111e, #head,max: APSN21: 04.01.2014 19:05:50
E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt end
17837:
17838: //Tag Macros
17839:
17840: asm #name, #date, #host, #path, #file, #head, #sign #tech #net end
17841:
17842: //Tag Macros
17843: 10188: SearchAndCopy(memol.lines, '#name', getUsernameWin, 11);
17844: 10189: SearchAndCopy(memol.lines, '#date', datetimetToStr(now), 11);

```

```

17845: 10190: SearchAndCopy(mem01.lines, '#host', getComputernameWin, 11);
17846: 10191: SearchAndCopy(mem01.lines, '#path', fpath, 11);
17847: 10192: SearchAndCopy(mem01.lines, '#file', fname, 11);
17848: 10199: SearchAndCopy(mem01.lines, '#fils', fname + ' ' + SHA1(Act_Filename), 11);
17849: 10193: SearchAndCopy(mem01.lines, '#locs', intToStr(getCodeEnd), 11);
17850: 10194: SearchAndCopy(mem01.lines, '#perf', perftime, 11);
17851: 10195: SearchAndCopy(mem01.lines, '#sign', Format('%s: %s: %s',
17852: [getUserNameWin, getComputernameWin, datetimetToStr(now),
17853: 10196: SearchAndCopy(mem01.lines, '#head', Format('%s: %s: %s %s ',
17854: 10197: [getUserNameWin, getComputernameWin, datetimetToStr(now), Act_Filename]), 11);
17855: [getUserNameWin, getComputernameWin, datetimetToStr(now), Act_Filename]), 11);
17856: 10198: SearchAndCopy(mem01.lines, '#tech', Format('perf: %s threads: %d %s %s',
17857: [perftime, numprocessthreads, getIPAddress(getComputerNameWin), timetoStr(time), mbversion]), 11);
17858: 10298: SearchAndCopy(mem01.lines, '#net', Format('DNS: %s; local IPs: %s; local IP: %s',
17859: [getDNS, GetLocalIPs, getIPAddress(getComputerNameWin)], 10);
17860:
17861: // #tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
17862:
17863: // Replace Macros
17864: SearchAndCopy(mem01.lines, '<TIME>', timetoStr(time), 6);
17865: SearchAndCopy(mem01.lines, '<DATE>', datetimetToStr(date), 6);
17866: SearchAndCopy(mem01.lines, '<PATH>', fpath, 6);
17867: SearchAndCopy(mem01.lines, '<EXEPATH>', EXEPath, 9);
17868: SearchAndCopy(mem01.lines, '<FILE>', fname, 6);
17869: SearchAndCopy(mem01.lines, '<SOURCE>', ExePath + 'Source', 8);
17870:
17871: 10198: SearchAndCopy(mem01.lines, '#tech!perf: threads: 2 192.168.1.53 19:05:50 3.9.9.84
17872: [perftime, numprocessthreads, getIPAddress(getComputerNameWin), timetoStr(time), mbversion]), 11);
17873: // #tech!84perf: threads: 2 192.168.1.53 19:05:50 3.9.9.84
17874: SearchAndCopy(mem01.lines, 'maxbox_extract_funclist399.txt
17875:
17876: //-----
17877: //*****mX4 ToDo List Tags ../Help/ToDo List*****
17878: //-----
17879:
17880: while I < sl.Count do begin
17881: // if MatchesMask(sl[I], '/*? TODO ([a-z0-9_]*#[1-9#]*:*)' then
17882: if MatchesMask(sl[I], '/*? TODO (?*#?#)*:*)' then
17883: BreakupToDo(Filename, sl, I, 'TODO', True, True) // full info TODO
17884: else if MatchesMask(sl[I], '/*? DONE (?*#?#)*:*)' then
17885: BreakupToDo(Filename, sl, I, 'DONE', True, True) // full info DONE
17886: else if MatchesMask(sl[I], '/*? TODO (#?#)*:*)' then
17887: BreakupToDo(Filename, sl, I, 'TODO', False, True) // only priority info TODO
17888: else if MatchesMask(sl[I], '/*? DONE (#?#)*:*)' then
17889: BreakupToDo(Filename, sl, I, 'DONE', False, True) // only priority info DONE
17890: else if MatchesMask(sl[I], '/*?*TODO*:*)' then
17891: BreakupToDo(Filename, sl, I, 'TODO', False, False) // custom TODO
17892: else if MatchesMask(sl[I], '/*?*DONE*:*)' then
17893: BreakupToDo(Filename, sl, I, 'DONE', False, False); // custom DONE
17894: Inc(I);
17895: end;
17896:
17897:
17898: //-----
17899: //*****mX4 Public Tools API *****
17900: //-----
17901: file : unit uPSI_fMain.pas; [SOTAP] Open Tools API Catalog
17902: // Those functions concern the editor and preprocessor, all of the IDE
17903: Example: Call it with maxform1.InfolClick(self)
17904: Note: Call all Methods with maxForm1., e.g.:
17905: maxForm1.ShellStyle1Click(self);
17906:
17907: procedure SIRegister_fMain(CL: TPSPascalCompiler);
17908: begin
17909: Const('BYTECODE', 'String').SetString( 'bytecode.txt
17910: Const('PSTEXT', 'String').SetString( 'PS Scriptfiles (*.txt)|*.TXT
17911: Const('PSMODEL', 'String').SetString( 'PS Modelfiles (*.uc)|*.UC
17912: Const('PSPASCAL', 'String').SetString( 'PS Pascalfiles (*.pas)|*.PAS
17913: Const('PSINC', 'String').SetString( 'PS Includes (*.inc)|*.INC
17914: Const('DEFFILENAME', 'String').SetString( 'firstdemo.txt
17915: Const('DEFINIFILE', 'String').SetString( 'maxboxdef.ini
17916: Const('EXCEPTLOGFILE', 'String').SetString( 'maxboxerrorlog.txt
17917: Const('ALLFUNCTIONSLIST', 'String').SetString( 'upsi_allfunctionslist.txt
17918: Const('ALLFUNCTIONSLISTPDF', 'String').SetString( 'maxbox_functions_all.pdf
17919: Const('ALLOBJECTSLIST', 'String').SetString( 'docs\VCL.pdf
17920: Const('ALLRESOURCELIST', 'String').SetString( 'docs\upsi_allresourceclist.txt
17921: Const('ALLUNITLIST', 'String').SetString( 'docs\maxbox3_9.xml');
17922: Const('INCLUDEBOX', 'String').SetString( 'pas_includebox.inc
17923: Const('BOOTSCRIPT', 'String').SetString( 'maxbootscript.txt
17924: Const('MBVERSION', 'String').SetString( '3.9.9.94
17925: Const('VERSION', 'String').SetString( '3.9.9.94
17926: Const('MBVER', 'String').SetString( '399
17927: Const('MBVERI', 'Integer').SetInt(399);
17928: Const('MBVERIALL', 'Integer').SetInt(39994);
17929: Const('EXENAME', 'String').SetString( 'maxbox3.exe
17930: Const('MXSITE', 'String').SetString( 'http://www.softwareschule.ch/maxbox.htm
17931: Const('MXVERSIONFILE', 'String').SetString( 'http://www.softwareschule.ch/maxvfile.txt
17932: Const('MXINTERNETCHECK', 'String').SetString( 'www.ask.com
17933: Const('MXMAIL', 'String').SetString( 'max@kleiner.com

```



```

17934:   Const('TAB','Char').SetString( #09);
17935:   Const('CODECOMPLETION','String').SetString( 'bds_delphi.dci
17936:   SIRegister_TMaxForm1(CL);
17937: end;
17938:
17939:   with FindClass('TForm'),'TMaxForm1' do begin
17940:     memo2, 'TMemo', iptrw);
17941:     memo1, 'TSynMemo', iptrw);
17942:     CB1SCList, 'TComboBox', iptrw);
17943:     mxNavigator, 'TComboBox', iptrw);
17944:     IPhost, 'string', iptrw);
17945:     IPort, 'integer', iptrw);
17946:     COMPort, 'integer', iptrw); //3.9.6.4
17947:     Splitter1, 'TSplitter', iptrw);
17948:     PSScript, 'TPSScript', iptrw);
17949:     PS3DllPlugin, 'TPSDllPlugin', iptrw);
17950:     MainMenu1, 'TMainMenu', iptrw);
17951:     Program1, 'TMenuItem', iptrw);
17952:     Compile1, 'TMenuItem', iptrw);
17953:     Files1, 'TMenuItem', iptrw);
17954:     open1, 'TMenuItem', iptrw);
17955:     Save2, 'TMenuItem', iptrw);
17956:     Options1, 'TMenuItem', iptrw);
17957:     Savebefore1, 'TMenuItem', iptrw);
17958:     Largefont1, 'TMenuItem', iptrw);
17959:     sBytecode1, 'TMenuItem', iptrw);
17960:     Saveas3, 'TMenuItem', iptrw);
17961:     Clear1, 'TMenuItem', iptrw);
17962:     Slinenumbers1, 'TMenuItem', iptrw);
17963:     About1, 'TMenuItem', iptrw);
17964:     Search1, 'TMenuItem', iptrw);
17965:     SynPasSyn1, 'TSynPasSyn', iptrw);
17966:     memo1, 'TSynMemo', iptrw);
17967:     SynEditSearch1, 'TSynEditSearch', iptrw);
17968:     WordWrap1, 'TMenuItem', iptrw);
17969:     XPManifest1, 'TXPManifest', iptrw);
17970:     SearchNext1, 'TMenuItem', iptrw);
17971:     Replace1, 'TMenuItem', iptrw);
17972:     PSImport_Controls1, 'TPSImport_Controls', iptrw);
17973:     PSImport_Classes1, 'TPSImport_Classes', iptrw);
17974:     ShowIncludel, 'TMenuItem', iptrw);
17975:     SynEditPrint1, 'TSynEditPrint', iptrw);
17976:     Printout1, 'TMenuItem', iptrw);
17977:     mnPrintColors1, 'TMenuItem', iptrw);
17978:     dlgFilePrint, 'TPrintDialog', iptrw);
17979:     dlgPrintFont1, 'TFontDialog', iptrw);
17980:     mnuPrintFont1, 'TMenuItem', iptrw);
17981:     Includel, 'TMenuItem', iptrw);
17982:     CodeCompletionList1, 'TMenuItem', iptrw);
17983:     IncludeList1, 'TMenuItem', iptrw);
17984:     ImageList1, 'TImageList', iptrw);
17985:     ImageList2, 'TImageList', iptrw);
17986:     CoolBar1, 'TCoolBar', iptrw);
17987:     ToolBar1, 'TToolBar', iptrw);
17988:     tbtnLoad, 'TToolButton', iptrw);
17989:     ToolButton2, 'TToolButton', iptrw);
17990:     tbtnFind, 'TToolButton', iptrw);
17991:     tbtnCompile, 'TToolButton', iptrw);
17992:     tbtnTrans, 'TToolButton', iptrw);
17993:     tbtnUseCase, 'TToolButton', iptrw); //3.8
17994:     toolbtnTutorial, 'TToolButton', iptrw);
17995:     tbtn6res, 'TToolButton', iptrw);
17996:     ToolButton5, 'TToolButton', iptrw);
17997:     ToolButton1, 'TToolButton', iptrw);
17998:     ToolButton3, 'TToolButton', iptrw);
17999:     statusBar1, 'TStatusBar', iptrw);
18000:     SaveOutput1, 'TMenuItem', iptrw);
18001:     ExportClipboard1, 'TMenuItem', iptrw);
18002:     Close1, 'TMenuItem', iptrw);
18003:     Manual1, 'TMenuItem', iptrw);
18004:     About2, 'TMenuItem', iptrw);
18005:     loadLastfile1, 'TMenuItem', iptrw);
18006:     imglogo, 'TImage', iptrw);
18007:     cedebug, 'TPSScriptDebugger', iptrw);
18008:     debugPopupMenu1, 'TPopupMenu', iptrw);
18009:     BreakPointMenu, 'TMenuItem', iptrw);
18010:     Decompile1, 'TMenuItem', iptrw);
18011:     StepIntol, 'TMenuItem', iptrw);
18012:     StepOut1, 'TMenuItem', iptrw);
18013:     Reset1, 'TMenuItem', iptrw);
18014:     DebugRun1, 'TMenuItem', iptrw);
18015:     PSImport_ComObj1, 'TPSImport_ComObj', iptrw);
18016:     PSImport_StdCtrls1, 'TPSImport_StdCtrls', iptrw);
18017:     PSImport_Forms1, 'TPSImport_Forms', iptrw);
18018:     PSImport_DateUtils1, 'TPSImport_DateUtils', iptrw);
18019:     tutorial4, 'TMenuItem', iptrw);
18020:     ExporttoClipboard1, 'TMenuItem', iptrw);
18021:     ImportfromClipboard1, 'TMenuItem', iptrw);
18022:     N4, 'TMenuItem', iptrw);

```

```
18023: N5', 'TMenuItem', iptrw);
18024: N6', 'TMenuItem', iptrw);
18025: ImportfromClipboard2', 'TMenuItem', iptrw);
18026: tutorial1', 'TMenuItem', iptrw);
18027: N7', 'TMenuItem', iptrw);
18028: ShowSpecChars1', 'TMenuItem', iptrw);
18029: OpenDirectory1', 'TMenuItem', iptrw);
18030: procMess', 'TMenuItem', iptrw);
18031: tbtnUseCase', 'TToolButton', iptrw);
18032: ToolButton7', 'TToolButton', iptrw);
18033: EditFont1', 'TMenuItem', iptrw);
18034: UseCase1', 'TMenuItem', iptrw);
18035: tutorial21', 'TMenuItem', iptrw);
18036: OpenUseCase1', 'TMenuItem', iptrw);
18037: PSImport_DB1', 'TPSImport_DB', iptrw);
18038: tutorial31', 'TMenuItem', iptrw);
18039: SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
18040: HTMLSyntax1', 'TMenuItem', iptrw);
18041: ShowInterfaces1', 'TMenuItem', iptrw);
18042: Tutorial5', 'TMenuItem', iptrw);
18043: AllFunctionsList1', 'TMenuItem', iptrw);
18044: ShowLastException1', 'TMenuItem', iptrw);
18045: PlayMP31', 'TMenuItem', iptrw);
18046: SynTeXSyn1', 'TSynTeXSyn', iptrw);
18047: texSyntax1', 'TMenuItem', iptrw);
18048: N8', 'TMenuItem', iptrw);
18049: GetEmails1', 'TMenuItem', iptrw);
18050: SynCppSyn1', 'TSynCppSyn', iptrw);
18051: CSyntax1', 'TMenuItem', iptrw);
18052: Tutorial6', 'TMenuItem', iptrw);
18053: New1', 'TMenuItem', iptrw);
18054: AllObjectsList1', 'TMenuItem', iptrw);
18055: LoadBytecode1', 'TMenuItem', iptrw);
18056: CipherFile1', 'TMenuItem', iptrw);
18057: N9', 'TMenuItem', iptrw);
18058: N10', 'TMenuItem', iptrw);
18059: Tutorial11', 'TMenuItem', iptrw);
18060: Tutorial71', 'TMenuItem', iptrw);
18061: UpdateService1', 'TMenuItem', iptrw);
18062: PascalsSchool1', 'TMenuItem', iptrw);
18063: Tutorial81', 'TMenuItem', iptrw);
18064: DelphiSite1', 'TMenuItem', iptrw);
18065: Output1', 'TMenuItem', iptrw);
18066: TerminalStyle1', 'TMenuItem', iptrw);
18067: ReadOnly1', 'TMenuItem', iptrw);
18068: ShellStyle1', 'TMenuItem', iptrw);
18069: BigScreen1', 'TMenuItem', iptrw);
18070: Tutorial91', 'TMenuItem', iptrw);
18071: SaveOutput2', 'TMenuItem', iptrw);
18072: N11', 'TMenuItem', iptrw);
18073: SaveScreenshot', 'TMenuItem', iptrw);
18074: Tutorial101', 'TMenuItem', iptrw);
18075: SQLSyntax1', 'TMenuItem', iptrw);
18076: SynSQLSyn1', 'TSynSQLSyn', iptrw);
18077: Console1', 'TMenuItem', iptrw);
18078: SynXMLSyn1', 'TSynXMLSyn', iptrw);
18079: XMLSyntax1', 'TMenuItem', iptrw);
18080: ComponentCount1', 'TMenuItem', iptrw);
18081: NewInstancel', 'TMenuItem', iptrw);
18082: toolbtnTutorial', 'TToolButton', iptrw);
18083: Memory1', 'TMenuItem', iptrw);
18084: SynJavaSyn1', 'TSynJavaSyn', iptrw);
18085: JavaSyntax1', 'TMenuItem', iptrw);
18086: SyntaxCheck1', 'TMenuItem', iptrw);
18087: Tutorial10Statistics1', 'TMenuItem', iptrw);
18088: ScriptExplorer1', 'TMenuItem', iptrw);
18089: FormOutput1', 'TMenuItem', iptrw);
18090: ArduinoDump1', 'TMenuItem', iptrw);
18091: AndroidDump1', 'TMenuItem', iptrw);
18092: GotoEnd1', 'TMenuItem', iptrw);
18093: AllResourceList1', 'TMenuItem', iptrw);
18094: ToolButton4', 'TToolButton', iptrw);
18095: tbtn6res', 'TToolButton', iptrw);
18096: Tutorial11Forms1', 'TMenuItem', iptrw);
18097: Tutorial12SQL1', 'TMenuItem', iptrw);
18098: ResourceExplore1', 'TMenuItem', iptrw);
18099: Infol', 'TMenuItem', iptrw);
18100: N12', 'TMenuItem', iptrw);
18101: CryptoBox1', 'TMenuItem', iptrw);
18102: Tutorial13Ciphering1', 'TMenuItem', iptrw);
18103: CipherFile2', 'TMenuItem', iptrw);
18104: N13', 'TMenuItem', iptrw);
18105: ModulesCount1', 'TMenuItem', iptrw);
18106: AddOns2', 'TMenuItem', iptrw);
18107: N4GewinntGame1', 'TMenuItem', iptrw);
18108: DocuforAddOns1', 'TMenuItem', iptrw);
18109: Tutorial14Async1', 'TMenuItem', iptrw);
18110: Lessons15Review1', 'TMenuItem', iptrw);
18111: SynPHPSyn1', 'TSynPHPSyn', iptrw);
```

```
18112: PHPSyntax1', 'TMenuItem', iptrw);
18113: Breakpoint1', 'TMenuItem', iptrw);
18114: SerialRS2321', 'TMenuItem', iptrw);
18115: N14', 'TMenuItem', iptrw);
18116: SynCSSyn1', 'TSynCSSyn', iptrw);
18117: CSyntax2', 'TMenuItem', iptrw);
18118: Calculator1', 'TMenuItem', iptrw);
18119: tbtnSerial', 'TToolButton', iptrw);
18120: ToolButton8', 'TToolButton', iptrw);
18121: Tutorial151', 'TMenuItem', iptrw);
18122: N15', 'TMenuItem', iptrw);
18123: N16', 'TMenuItem', iptrw);
18124: ControlBar1', 'TControlBar', iptrw);
18125: ToolBar2', 'TToolBar', iptrw);
18126: BtnOpen', 'TToolButton', iptrw);
18127: BtnSave', 'TToolButton', iptrw);
18128: BtnPrint', 'TToolButton', iptrw);
18129: BtnColors', 'TToolButton', iptrw);
18130: btnClassReport', 'TToolButton', iptrw);
18131: BtnRotateRight', 'TToolButton', iptrw);
18132: BtnFullSize', 'TToolButton', iptrw);
18133: BtnFitToWindowSize', 'TToolButton', iptrw);
18134: BtnZoomMinus', 'TToolButton', iptrw);
18135: BtnZoomPlus', 'TToolButton', iptrw);
18136: Panell1', 'TPanel', iptrw);
18137: LabelBrettgroesse', 'TLabel', iptrw);
18138: CB1SCList', 'TComboBox', iptrw);
18139: ImageListNormal', 'TImageList', iptrw);
18140: spbtnexplore', 'TSpeedButton', iptrw);
18141: spbtnexample', 'TSpeedButton', iptrw);
18142: spbsaveas', 'TSpeedButton', iptrw);
18143: imglogobox', 'TImage', iptrw);
18144: EnlargeFont1', 'TMenuItem', iptrw);
18145: EnlargeFont2', 'TMenuItem', iptrw);
18146: ShrinkFont1', 'TMenuItem', iptrw);
18147: ThreadDemol', 'TMenuItem', iptrw);
18148: HEXEditor1', 'TMenuItem', iptrw);
18149: HEXView1', 'TMenuItem', iptrw);
18150: HEXInspect1', 'TMenuItem', iptrw);
18151: SynExporterHTML1', 'TSynExporterHTML', iptrw);
18152: ExporttoHTML1', 'TMenuItem', iptrw);
18153: ClassCount1', 'TMenuItem', iptrw);
18154: HTMLOutput1', 'TMenuItem', iptrw);
18155: HEXEditor2', 'TMenuItem', iptrw);
18156: Minesweeper1', 'TMenuItem', iptrw);
18157: N17', 'TMenuItem', iptrw);
18158: PicturePuzzle1', 'TMenuItem', iptrw);
18159: sbvclhelp', 'TSpeedButton', iptrw);
18160: DependencyWalker1', 'TMenuItem', iptrw);
18161: WebScanner1', 'TMenuItem', iptrw);
18162: View1', 'TMenuItem', iptrw);
18163: mnToolbar1', 'TMenuItem', iptrw);
18164: mnStatusbar2', 'TMenuItem', iptrw);
18165: mnConsole2', 'TMenuItem', iptrw);
18166: mnCoolbar2', 'TMenuItem', iptrw);
18167: mnSplitter2', 'TMenuItem', iptrw);
18168: WebServer1', 'TMenuItem', iptrw);
18169: Tutorial17Server1', 'TMenuItem', iptrw);
18170: Tutorial18Arduinol', 'TMenuItem', iptrw);
18171: SynPerlSyn1', 'TSynPerlSyn', iptrw);
18172: PerlSyntax1', 'TMenuItem', iptrw);
18173: SynPythonSyn1', 'TSynPythonSyn', iptrw);
18174: PythonSyntax1', 'TMenuItem', iptrw);
18175: DMathLibrary1', 'TMenuItem', iptrw);
18176: IntfNavigator1', 'TMenuItem', iptrw);
18177: EnlargeFontConsole1', 'TMenuItem', iptrw);
18178: ShrinkFontConsole1', 'TMenuItem', iptrw);
18179: SetInterfaceList1', 'TMenuItem', iptrw);
18180: popintfList', 'TPopupMenu', iptrw);
18181: intfAdd1', 'TMenuItem', iptrw);
18182: intfDelete1', 'TMenuItem', iptrw);
18183: intfRefactor1', 'TMenuItem', iptrw);
18184: Defactor1', 'TMenuItem', iptrw);
18185: Tutorial19COMArduinol', 'TMenuItem', iptrw);
18186: Tutorial20Regex', 'TMenuItem', iptrw);
18187: N18', 'TMenuItem', iptrw);
18188: ManualE1', 'TMenuItem', iptrw);
18189: FullTextFinder1', 'TMenuItem', iptrw);
18190: Move1', 'TMenuItem', iptrw);
18191: FractalDemol', 'TMenuItem', iptrw);
18192: Tutorial21Android1', 'TMenuItem', iptrw);
18193: Tutorial0Function1', 'TMenuItem', iptrw);
18194: SimuLogBox1', 'TMenuItem', iptrw);
18195: OpenExamples1', 'TMenuItem', iptrw);
18196: SynJScriptSyn1', 'TSynJScriptSyn', iptrw);
18197: JavaScriptSyntax1', 'TMenuItem', iptrw);
18198: Halt1', 'TMenuItem', iptrw);
18199: CodeSearch1', 'TMenuItem', iptrw);
18200: SynRubySyn1', 'TSynRubySyn', iptrw);
```

```

18201: RubySyntax1', 'TMenuItem', iptrw);
18202: Undo1', 'TMenuItem', iptrw);
18203: SynUNIXShellScriptSyn1', 'TSynUNIXShellScriptSyn', iptrw);
18204: LinuxShellScript1', 'TMenuItem', iptrw);
18205: Rename1', 'TMenuItem', iptrw);
18206: spdcodesearch', 'TSpeedButton', iptrw);
18207: Preview1', 'TMenuItem', iptrw);
18208: Tutorial22Services1', 'TMenuItem', iptrw);
18209: Tutorial23RealTime1', 'TMenuItem', iptrw);
18210: Configuration1', 'TMenuItem', iptrw);
18211: MP3Player1', 'TMenuItem', iptrw);
18212: DLLSpy1', 'TMenuItem', iptrw);
18213: SynURIOpener1', 'TSynURIOpener', iptrw);
18214: SynURISyn1', 'TSynURISyn', iptrw);
18215: URILinksClicks1', 'TMenuItem', iptrw);
18216: EditReplace1', 'TMenuItem', iptrw);
18217: GotoLine1', 'TMenuItem', iptrw);
18218: ActiveLineColor1', 'TMenuItem', iptrw);
18219: ConfigFile1', 'TMenuItem', iptrw);
18220: SortIntfList1', 'TMenuItem', iptrw);
18221: Redo1', 'TMenuItem', iptrw);
18222: Tutorial24CleanCode1', 'TMenuItem', iptrw);
18223: Tutorial25Configuration1', 'TMenuItem', iptrw);
18224: IndentSelection1', 'TMenuItem', iptrw);
18225: UnindentSection1', 'TMenuItem', iptrw);
18226: SkyStyle1', 'TMenuItem', iptrw);
18227: N19', 'TMenuItem', iptrw);
18228: CountWords1', 'TMenuItem', iptrw);
18229: imbookmarkimages', 'TImageList', iptrw);
18230: Bookmark11', 'TMenuItem', iptrw);
18231: N20', 'TMenuItem', iptrw);
18232: Bookmark21', 'TMenuItem', iptrw);
18233: Bookmark31', 'TMenuItem', iptrw);
18234: Bookmark41', 'TMenuItem', iptrw);
18235: SynMultiSyn1', 'TSynMultiSyn', iptrw);
18236:
18237: Procedure IFPS3ClassesPluginCompImport( Sender : TObject; x : TPSPascalCompiler)
18238: Procedure IFPS3ClassesPluginExecImport(Sender: TObject; Exec: TPSExec; x:TPSRuntimeClassImporter);
18239: Procedure PSScriptCompile( Sender : TPSScript)
18240: Procedure Compile1Click( Sender : TObject)
18241: Procedure PSScriptExecute( Sender : TPSScript)
18242: Procedure open1Click( Sender : TObject)
18243: Procedure Save2Click( Sender : TObject)
18244: Procedure Savebefore1Click( Sender : TObject)
18245: Procedure Largefont1Click( Sender : TObject)
18246: Procedure FormActivate( Sender : TObject)
18247: Procedure SBytecode1Click( Sender : TObject)
18248: Procedure FormKeyPress( Sender : TObject; var Key : Char)
18249: Procedure Saveas3Click( Sender : TObject)
18250: Procedure Clear1Click( Sender : TObject)
18251: Procedure Slinenumbers1Click( Sender : TObject)
18252: Procedure About1Click( Sender : TObject)
18253: Procedure Search1Click( Sender : TObject)
18254: Procedure FormCreate( Sender : TObject)
18255: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Integer;
18256: var Action : TSynReplaceAction)
18257: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
18258: Procedure WordWrap1Click( Sender : TObject)
18259: Procedure SearchNext1Click( Sender : TObject)
18260: Procedure Replace1Click( Sender : TObject)
18261: Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FName,Output:String):Bool;
18262: Procedure ShowInclude1Click( Sender : TObject)
18263: Procedure Printout1Click( Sender : TObject)
18264: Procedure mnuPrintFont1Click( Sender : TObject)
18265: Procedure Include1Click( Sender : TObject)
18266: Procedure FormDestroy( Sender : TObject)
18267: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
18268: Procedure UpdateView1Click( Sender : TObject)
18269: Procedure CodeCompletionList1Click( Sender : TObject)
18270: Procedure SaveOutput1Click( Sender : TObject)
18271: Procedure ExportClipboard1Click( Sender : TObject)
18272: Procedure Close1Click( Sender : TObject)
18273: Procedure Manual1Click( Sender : TObject)
18274: Procedure LoadLastFile1Click( Sender : TObject)
18275: Procedure Memo1Change( Sender : TObject)
18276: Procedure Decompile1Click( Sender : TObject)
18277: Procedure StepInto1Click( Sender : TObject)
18278: Procedure StepOut1Click( Sender : TObject)
18279: Procedure Reset1Click( Sender : TObject)
18280: Procedure cedebugAfterExecute( Sender : TPSScript)
18281: Procedure cedebugBreakpoint(Sender:TObject; const FileName:String; Position,Row, Col: Cardinal)
18282: Procedure cedebugCompile( Sender : TPSScript)
18283: Procedure cedebugExecute( Sender : TPSScript)
18284: Procedure cedebugIdle( Sender : TObject)
18285: Procedure cedebugLineInfo( Sender:TObject;const FileName:String; Position, Row, Col : Cardinal)
18286: Procedure Memo1SpecialLineColors(Sender: TObject; Line:Int; var Special:Boolean;var FG,BG:TColor);
18287: Procedure BreakPointMenuClick( Sender : TObject)
18288: Procedure DebugRun1Click( Sender : TObject)
18289: Procedure tutorial4Click( Sender : TObject)

```



```

18290: Procedure ImportfromClipboard1Click( Sender : TObject)
18291: Procedure ImportfromClipboard2Click( Sender : TObject)
18292: Procedure tutorial1Click( Sender : TObject)
18293: Procedure ShowSpecChars1Click( Sender : TObject)
18294: Procedure StatusBar1Db1Click( Sender : TObject)
18295: Procedure PSScriptLine( Sender : TObject)
18296: Procedure OpenDirectory1Click( Sender : TObject)
18297: Procedure procMessClick( Sender : TObject)
18298: Procedure tbtnUseCaseClick( Sender : TObject)
18299: Procedure EditFont1Click( Sender : TObject)
18300: Procedure tutorial21Click( Sender : TObject)
18301: Procedure tutorial31Click( Sender : TObject)
18302: Procedure HTMLSyntax1Click( Sender : TObject)
18303: Procedure ShowInterfaces1Click( Sender : TObject)
18304: Procedure Tutorial5Click( Sender : TObject)
18305: Procedure ShowLastException1Click( Sender : TObject)
18306: Procedure PlayMP31Click( Sender : TObject)
18307: Procedure AllFunctionsList1Click( Sender : TObject)
18308: Procedure texSyntax1Click( Sender : TObject)
18309: Procedure GetEMails1Click( Sender : TObject)
18310: procedure DelphiSite1Click(Sender: TObject);
18311: procedure TerminalStyle1Click(Sender: TObject);
18312: procedure ReadOnly1Click(Sender: TObject);
18313: procedure ShellStyle1Click(Sender: TObject);
18314: procedure Console1Click(Sender: TObject); //3.2
18315: procedure BigScreen1Click(Sender: TObject);
18316: procedure Tutorial91Click(Sender: TObject);
18317: procedure SaveScreenshotClick(Sender: TObject);
18318: procedure Tutorial101Click(Sender: TObject);
18319: procedure SQLSyntax1Click(Sender: TObject);
18320: procedure XMLSyntax1Click(Sender: TObject);
18321: procedure ComponentCount1Click(Sender: TObject);
18322: procedure NewInstance1Click(Sender: TObject);
18323: procedure CSyntax1Click(Sender: TObject);
18324: procedure Tutorial6Click(Sender: TObject);
18325: procedure New1Click(Sender: TObject);
18326: procedure AllObjectsList1Click(Sender: TObject);
18327: procedure LoadBytecode1Click(Sender: TObject);
18328: procedure CipherFile1Click(Sender: TObject); //V3.5
18329: procedure NewInstance1Click(Sender: TObject);
18330: procedure toolbtnTutorialClick(Sender: TObject);
18331: procedure Memory1Click(Sender: TObject);
18332: procedure JavaSyntax1Click(Sender: TObject);
18333: procedure SyntaxCheck1Click(Sender: TObject);
18334: procedure ScriptExplorer1Click(Sender: TObject);
18335: procedure FormOutput1Click(Sender: TObject); //V3.6
18336: procedure GotoEnd1Click(Sender: TObject);
18337: procedure AllResourceList1Click(Sender: TObject);
18338: procedure tbtn6resClick(Sender: TObject); //V3.7
18339: procedure Info1Click(Sender: TObject);
18340: procedure Tutorial10Statistics1Click(Sender: TObject);
18341: procedure Tutorial11Forms1Click(Sender: TObject);
18342: procedure Tutorial12SQL1Click(Sender: TObject); //V3.8
18343: procedure ResourceExplore1Click(Sender: TObject);
18344: procedure Info1Click(Sender: TObject);
18345: procedure CryptoBox1Click(Sender: TObject);
18346: procedure ModulesCount1Click(Sender: TObject);
18347: procedure N4GewinntGame1Click(Sender: TObject);
18348: procedure PHPSyntax1Click(Sender: TObject);
18349: procedure SerialRS2321Click(Sender: TObject);
18350: procedure CSyntax2Click(Sender: TObject);
18351: procedure Calculator1Click(Sender: TObject);
18352: procedure Tutorial13Ciphering1Click(Sender: TObject);
18353: procedure Tutorial14Async1Click(Sender: TObject);
18354: procedure PHPSyntax1Click(Sender: TObject);
18355: procedure BtnZoomPlusClick(Sender: TObject);
18356: procedure BtnZoomMinusClick(Sender: TObject);
18357: procedure btnClassReportClick(Sender: TObject);
18358: procedure ThreadDemo1Click(Sender: TObject);
18359: procedure HEXView1Click(Sender: TObject);
18360: procedure ExporttoHTML1Click(Sender: TObject);
18361: procedure Minesweeper1Click(Sender: TObject);
18362: procedure PicturePuzzle1Click(Sender: TObject); //V3.9
18363: procedure sbvclhelpClick(Sender: TObject);
18364: procedure DependencyWalker1Click(Sender: TObject);
18365: procedure CB1SCListDrawItem(Control: TWinControl; Index: Int; aRect: TRect; State: TOwnerDrawState);
18366: procedure WebScanner1Click(Sender: TObject);
18367: procedure mnToolBar1Click(Sender: TObject);
18368: procedure mnStatusBar2Click(Sender: TObject);
18369: procedure mnConsole2Click(Sender: TObject);
18370: procedure mnCoolbar2Click(Sender: TObject);
18371: procedure mnSplitter2Click(Sender: TObject);
18372: procedure WebServer1Click(Sender: TObject);
18373: procedure PerlSyntax1Click(Sender: TObject);
18374: procedure PythonSyntax1Click(Sender: TObject);
18375: procedure DMathLibrary1Click(Sender: TObject);
18376: procedure IntfNavigator1Click(Sender: TObject);
18377: procedure FullTextFinder1Click(Sender: TObject);
18378: function AppName: string;

```

```

18379: function ScriptName: string;
18380: function LastName: string;
18381: procedure FractalDemo1Click(Sender: TObject);
18382: procedure SimuLogBox1Click(Sender: TObject);
18383: procedure OpenExamples1Click(Sender: TObject);
18384: procedure Halt1Click(Sender: TObject);
18385: procedure Stop;
18386: procedure CodeSearch1Click(Sender: TObject);
18387: procedure RubySyntax1Click(Sender: TObject);
18388: procedure Undo1Click(Sender: TObject);
18389: procedure LinuxShellScript1Click(Sender: TObject);
18390: procedure WebScannerDirect(urls: string);
18391: procedure WebScanner(urls: string);
18392: procedure LoadInterfaceList2;
18393: procedure DLLSpy1Click(Sender: TObject);
18394: procedure Memo1Db1Click(Sender: TObject);
18395: procedure URILinksClicks1Click(Sender: TObject);
18396: procedure GotoLine1Click(Sender: TObject);
18397: procedure ConfigFile1Click(Sender: TObject);
18398: procedure Sort1IntfListClick(Sender: TObject);
18399: procedure Redo1Click(Sender: TObject);
18400: procedure Tutorial24CleanCode1Click(Sender: TObject);
18401: procedure IndentSelection1Click(Sender: TObject);
18402: procedure UnindentSection1Click(Sender: TObject);
18403: procedure SkyStyle1Click(Sender: TObject);
18404: procedure CountWords1Click(Sender: TObject);
18405: procedure Memo1PlaceBookmark(Sender: TObject; var Mark: TSynEditMark);
18406: procedure Memo1GutterClick(Sender: TObject; Button: TMouseButton; X, Y, Line: Integer; Mark: TSynEditMark);
18407: procedure Bookmark11Click(Sender: TObject);
18408: procedure Bookmark21Click(Sender: TObject);
18409: procedure Bookmark31Click(Sender: TObject);
18410: procedure Bookmark41Click(Sender: TObject);
18411: procedure SynMultiSyn1CustomRange(Sender: TSynMultiSyn; Operation: TRangeOperation; var Range: Pointer);
18412: 'STATMemoryReport', 'boolean', iptrw);
18413: 'IPPort', 'integer', iptrw);
18414: 'COMPort', 'integer', iptrw);
18415: 'lbintfList', 'TListBox', iptrw);
18416: Function GetStatChange: boolean
18417: Procedure SetStatChange(vstat: boolean)
18418: Function GetActFileName: string
18419: Procedure SetActFileName(vname: string)
18420: Function GetLastFileName: string
18421: Procedure SetLastFileName(vname: string)
18422: Procedure WebScannerDirect(urls: string)
18423: Procedure LoadInterfaceList2
18424: Function GetStatExecuteShell: boolean
18425: Procedure DoEditorExecuteCommand(EditorCommand: word)
18426: function GetActiveLineColor: TColor
18427: procedure SetActiveLineColor(acolor: TColor)
18428: procedure ScriptListBox1Click(Sender: TObject);
18429: procedure Memo2KeyPress(Sender: TObject; var Key: Char);
18430: procedure EnlargeGutter1Click(Sender: TObject);
18431: procedure Tetris1Click(Sender: TObject);
18432: procedure ToDoList1Click(Sender: TObject);
18433: procedure ProcessList1Click(Sender: TObject);
18434: procedure MetricReport1Click(Sender: TObject);
18435: procedure ProcessList1Click(Sender: TObject);
18436: procedure TCPSockets1Click(Sender: TObject);
18437: procedure ConfigUpdate1Click(Sender: TObject);
18438: procedure ADOWorkbench1Click(Sender: TObject);
18439: procedure SocketServer1Click(Sender: TObject);
18440: procedure FormDemo1Click(Sender: TObject);
18441: procedure Richedit1Click(Sender: TObject);
18442: procedure SimpleBrowser1Click(Sender: TObject);
18443: procedure DOSShell1Click(Sender: TObject);
18444: procedure SynExport1Click(Sender: TObject);
18445: procedure ExporttoRTF1Click(Sender: TObject);
18446: procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
18447: procedure SOAPTester1Click(Sender: TObject);
18448: procedure Sniffer1Click(Sender: TObject);
18449: procedure AutoDetectSyntax1Click(Sender: TObject);
18450: procedure FPlot1Click(Sender: TObject);
18451: procedure PasStyle1Click(Sender: TObject);
18452: procedure Tutorial183RGBLED1Click(Sender: TObject);
18453: procedure ReversilClick(Sender: TObject);
18454: procedure ManualmaXbox1Click(Sender: TObject);
18455: procedure BlaisePascalMagazine1Click(Sender: TObject);
18456: procedure AddToDo1Click(Sender: TObject);
18457: procedure CreateGUID1Click(Sender: TObject);
18458: procedure Tutorial27XML1Click(Sender: TObject);
18459: procedure CreateDLLStub1Click(Sender: TObject);
18460: procedure Tutorial28DLL1Click(Sender: TObject);
18461: procedure ResetKeyPressed;
18462: procedure FileChanges1Click(Sender: TObject);
18463: procedure OpenGLTry1Click(Sender: TObject);
18464: procedure AllUnitList1Click(Sender: TObject);
18465:
18466:
18467: //-----

```

```

18468: //*****mX4 Editor SynEdit Tools API *****
18469: //-----
18470: (*-----*)
18471: procedure SIRegister_TCustomSynEdit(CL: TPSPascalCompiler);
18472: begin
18473:   //with RegClassS(CL,'TCustomControl', 'TCustomSynEdit') do
18474:   with FindClass('TCustomControl'),'TCustomSynEdit' do begin
18475:     Constructor Create( AOwner : TComponent)
18476:       SelStart', 'Integer', iptrw);
18477:     SelEnd', 'Integer', iptrw); AlwaysShowCaret', 'Boolean', iptrw);
18478:     Procedure UpdateCaret
18479:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2:word;SS2:TShiftState);
18480:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2: word;SS2:TShiftState);
18481:     Procedure BeginUndoBlock
18482:     Procedure BeginUpdate
18483:     Function CaretInView : Boolean
18484:     Function CharIndexToRowCol( Index : integer) : TBufferCoord
18485:     Procedure Clear
18486:     Procedure ClearAll
18487:     Procedure ClearBookMark( BookMark : Integer)
18488:     Procedure ClearSelection
18489:     Procedure CommandProcessor( Command : TSynEditorCommand; AChar : char; Data : pointer)
18490:     Procedure ClearUndo
18491:     Procedure CopyToClipboard
18492:     Procedure CutToClipboard
18493:     Procedure DoCopyToClipboard( const SText : string)
18494:     Procedure EndUndoBlock
18495:     Procedure EndUpdate
18496:     Procedure EnsureCursorPosVisible
18497:     Procedure EnsureCursorPosVisibleEx( ForceToMiddle : Boolean)
18498:     Procedure FindMatchingBracket
18499:     Function GetMatchingBracket : TBufferCoord
18500:     Function GetMatchingBracketEx( const APoint : TBufferCoord) : TBufferCoord
18501:     Procedure ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer)
18502:     Function GetBookMark( BookMark : integer; var X, Y : integer) : boolean
18503:     Function GetHighlighterAttriAtRowCol( const XY: TBufferCoord; var Token : string; var Attri
18504:       : TSynHighlighterAttributes) : boolean
18505:     Function GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token : string;
18506:       var TokenType, Start : Integer; var Attri:TSynHighlighterAttributes):boolean
18507:     Function GetPositionOfMouse( out aPos : TBufferCoord) : Boolean
18508:     Function GetWordAtRowCol( const XY : TBufferCoord) : string
18509:     Procedure GotoBookMark( BookMark : Integer)
18510:     Procedure GotoLineAndCenter( ALine : Integer)
18511:     Function IdentChars : TSynIdentChars
18512:     Procedure InvalidateGutter
18513:     Procedure InvalidateGutterLine( aLine : integer)
18514:     Procedure InvalidateGutterLines( FirstLine, LastLine : integer)
18515:     Procedure InvalidateLine( Line : integer)
18516:     Procedure InvalidateLines( FirstLine, LastLine : integer)
18517:     Procedure InvalidateSelection
18518:     Function IsBookmark( BookMark : integer) : boolean
18519:     Function IsPointInSelection( const Value : TBufferCoord) : boolean
18520:     Procedure LockUndo
18521:     Function BufferToDisplayPos( const p : TBufferCoord) : TDisplayCoord
18522:     Function DisplayToBufferPos( const p : TDisplayCoord) : TBufferCoord
18523:     Function LineToRow( aLine : integer) : integer
18524:     Function RowToLine( aRow : integer) : integer
18525:     Function NextWordPos : TBufferCoord
18526:     Function NextWordPosEx( const XY : TBufferCoord) : TBufferCoord
18527:     Procedure PasteFromClipboard
18528:     Function WordStart : TBufferCoord
18529:     Function WordStartEx( const XY : TBufferCoord) : TBufferCoord
18530:     Function WordEnd : TBufferCoord
18531:     Function WordEndEx( const XY : TBufferCoord) : TBufferCoord
18532:     Function PrevWordPos : TBufferCoord
18533:     Function PrevWordPosEx( const XY : TBufferCoord) : TBufferCoord
18534:     Function PixelsToRowColumn( aX, aY : integer) : TDisplayCoord
18535:     Function PixelsToNearestRowColumn( aX, aY : integer) : TDisplayCoord
18536:     Procedure Redo
18537:     Procedure RegisterCommandHandler(const AHandlerProc:THookedCommandEvent;AHandlerData:pointer));
18538:     Function RowColumnToPixels( const RowCol : TDisplayCoord) : TPoint
18539:     Function RowColToCharIndex( RowCol : TBufferCoord) : integer
18540:     Function SearchReplace( const ASearch, AReplace:string; AOptions:TSynSearchOptions): integer
18541:     Procedure SelectAll
18542:     Procedure SetBookMark( BookMark : Integer; X : Integer; Y : Integer)
18543:     Procedure SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord)
18544:     Procedure SetDefaultKeystrokes
18545:     Procedure SetSelWord
18546:     Procedure SetWordBlock( Value : TBufferCoord)
18547:     Procedure Undo
18548:     Procedure UnlockUndo
18549:     Procedure UnregisterCommandHandler( AHandlerProc : THookedCommandEvent)
18550:     Procedure AddKeyUpHandler( aHandler : TKeyEvent)
18551:     Procedure RemoveKeyUpHandler( aHandler : TKeyEvent)
18552:     Procedure AddKeyDownHandler( aHandler : TKeyEvent)
18553:     Procedure RemoveKeyDownHandler( aHandler : TKeyEvent)
18554:     Procedure AddKeyPressHandler( aHandler : TKeyPressEvent)
18555:     Procedure RemoveKeyPressHandler( aHandler : TKeyPressEvent)
18556:     Procedure AddFocusControl( aControl : TWinControl)

```

```

18557: Procedure RemoveFocusControl( aControl : TWinControl)
18558: Procedure AddMouseDownHandler( aHandler : TMouseEvent)
18559: Procedure RemoveMouseDownHandler( aHandler : TMouseEvent)
18560: Procedure AddMouseUpHandler( aHandler : TMouseEvent)
18561: Procedure RemoveMouseUpHandler( aHandler : TMouseEvent)
18562: Procedure AddMouseCursorHandler( aHandler : TMouseEvent)
18563: Procedure RemoveMouseCursorHandler( aHandler : TMouseEvent)
18564: Procedure SetLinesPointer( ASynEdit : TCustomSynEdit)
18565: Procedure RemoveLinesPointer
18566: Procedure HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList)
18567: Procedure UnHookTextBuffer
18568: BlockBegin, 'TBufferCoord', iptrw);
18569: BlockEnd, 'TBufferCoord', iptrw);
18570: CanPaste, 'Boolean', iptr);
18571: CanRedo, 'boolean', iptr);
18572: CanUndo, 'boolean', iptr);
18573: CaretX, 'Integer', iptrw);
18574: CaretY, 'Integer', iptrw);
18575: CaretXY, 'TBufferCoord', iptrw);
18576: ActiveLineColor, 'TColor', iptrw);
18577: DisplayX, 'Integer', iptr);
18578: DisplayY, 'Integer', iptr);
18579: DisplayXY, 'TDisplayCoord', iptr);
18580: DisplayLineCount, 'integer', iptr);
18581: CharsInWindow, 'Integer', iptr);
18582: CharWidth, 'integer', iptr);
18583: Font, 'TFont', iptrw);
18584: GutterWidth, 'Integer', iptr);
18585: Highlighter, 'TSynCustomHighlighter', iptrw);
18586: LeftChar, 'Integer', iptrw);
18587: LineHeight, 'integer', iptr);
18588: LinesInWindow, 'Integer', iptr);
18589: LineText, 'string', iptrw);
18590: Lines, 'TStrings', iptrw);
18591: Marks, 'TSynEditMarkList', iptr);
18592: MaxScrollWidth, 'integer', iptrw);
18593: Modified, 'Boolean', iptrw);
18594: PaintLock, 'Integer', iptr);
18595: ReadOnly, 'Boolean', iptrw);
18596: SearchEngine, 'TSynEditSearchCustom', iptrw);
18597: SelAvail, 'Boolean', iptr); SelLength, 'integer', iptrw);
18598: SelTabBlock, 'Boolean', iptr);
18599: SelTabLine, 'Boolean', iptr);
18600: SelText, 'string', iptrw);
18601: StateFlags, 'TSynStateFlags', iptr);
18602: Text, 'string', iptrw);
18603: TopLine, 'Integer', iptrw);
18604: WordAtCursor, 'string', iptr);
18605: WordAtMouse, 'string', iptr);
18606: UndoList, 'TSynEditUndoList', iptr);
18607: RedoList, 'TSynEditUndoList', iptr);
18608: OnProcessCommand, 'TProcessCommandEvent', iptrw);
18609: BookmarkOptions, 'TSynBookMarkOpt', iptrw);
18610: BorderStyle, 'TSynBorderStyle', iptrw);
18611: ExtraLineSpacing, 'integer', iptrw);
18612: Gutter, 'TSynGutter', iptrw);
18613: HideSelection, 'boolean', iptrw);
18614: InsertCaret, 'TSynEditCaretType', iptrw);
18615: InsertMode, 'boolean', iptrw);
18616: IsScrolling, 'Boolean', iptr);
18617: Keystrokes, 'TSynEditKeyStrokes', iptrw);
18618: MaxUndo, 'Integer', iptrw);
18619: Options, 'TSynEditorOptions', iptrw);
18620: OverwriteCaret, 'TSynEditCaretType', iptrw);
18621: RightEdge, 'Integer', iptrw); RightEdgeColor, 'TColor', iptrw);
18622: ScrollHintColor, 'TColor', iptrw);
18623: ScrollHintFormat, 'TScrollHintFormat', iptrw);
18624: ScrollBars, 'TScrollStyle', iptrw);
18625: SelectedColor, 'TSynSelectedColor', iptrw);
18626: SelectionMode, 'TSynSelectionMode', iptrw);
18627: ActiveSelectionMode, 'TSynSelectionMode', iptrw);
18628: TabWidth, 'integer', iptrw); WantReturns, 'boolean', iptrw);
18629: WantTabs, 'boolean', iptrw); WordWrap, 'boolean', iptrw);
18630: WordWrapGlyph, 'TSynGlyph', iptrw);
18631: OnChange, 'TNotifyEvent', iptrw);
18632: OnClearBookmark, 'TPlaceMarkEvent', iptrw);
18633: OnCommandProcessed, 'TProcessCommandEvent', iptrw);
18634: OnContextHelp, 'TContextHelpEvent', iptrw);
18635: OnDropFiles, 'TDropFilesEvent', iptrw);
18636: OnGutterClick, 'TGutterClickEvent', iptrw);
18637: OnGutterGetText, 'TGutterGetTextEvent', iptrw);
18638: OnGutterPaint, 'TGutterPaintEvent', iptrw);
18639: OnMouseCursor, 'TMouseEvent', iptrw);
18640: OnPaint, 'TPaintEvent', iptrw);
18641: OnPlaceBookmark, 'TPlaceMarkEvent', iptrw);
18642: OnProcessUserCommand, 'TProcessCommandEvent', iptrw);
18643: OnReplaceText, 'TReplaceTextEvent', iptrw);
18644: OnSpecialLineColors, 'TSpecialLineColorsEvent', iptrw);
18645: OnStatusChange, 'TStatusChangeEvent', iptrw);

```



```

18646:   OnPaintTransient', 'TPaintTransient', iptrw);
18647:   OnScroll', 'TScrollEvent', iptrw);
18648: end;
18649: Procedure RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)
18650: Function GetPlaceableHighlighters : TSynHighlighterList
18651: Function EditorCommandToDescrString( Cmd : TSynEditorCommand) : string
18652: Function EditorCommandToCodeString( Cmd : TSynEditorCommand) : string
18653: Procedure GetEditorCommandValues( Proc : TGetStrProc)
18654: Procedure GetEditorCommandExtended( Proc : TGetStrProc)
18655: Function IdentToEditorCommand( const Ident : string; var Cmd : longint) : boolean
18656: Function EditorCommandToIdent( Cmd : longint; var Ident : string) : boolean
18657: Function ConvertCodeStringToExtended( AString : String) : String
18658: Function ConvertExtendedToCodeString( AString : String) : String
18659: Function ConvertExtendedToCommand( AString : String) : TSynEditorCommand
18660: Function ConvertCodeStringToCommand( AString : String) : TSynEditorCommand
18661: Function IndexToEditorCommand( const AIndex : Integer) : Integer
18662:
18663: TSynEditorOption = (
18664:   eoAltSetsColumnMode,           //Holding down the Alt Key will put the selection mode into columnar format
18665:   eoAutoIndent,                 //Will indent caret on newlines with same amount of leading whitespace as
18666:                                 // preceding line
18667:   eoAutoSizeMaxScrollWidth,     //Automatically resizes the MaxScrollWidth property when inserting text
18668:   eoDisableScrollArrows,        //Disables the scroll bar arrow buttons when you can't scroll in that
18669:                                 //direction any more
18670:   eoDragDropEditing,            //Allows to select a block of text and drag it within document to another
18671:                                 // location
18672:   eoDropFiles,                  //Allows the editor accept OLE file drops
18673:   eoEnhanceHomeKey,              //enhances home key positioning, similar to visual studio
18674:   eoEnhanceEndKey,              //enhances End key positioning, similar to JDeveloper
18675:   eoGroupUndo,                  //When undoing/redoning actions, handle all continous changes the same kind
18676:                                 // in one call
18677:                                 //instead undoing/redoning each command separately
18678:   eoHalfPageScroll,            //When scrolling with page-up and page-down commands, only scroll a half
18679:                                 //page at a time
18680:   eoHideShowScrollbars,         //if enabled, then scrollbars will only show if necessary.
18681:   If you have ScrollPastEOL,    //then it the horizontal bar will always be there (it uses MaxLength instead)
18682:   eoKeepCaretX,                 //When moving through lines w/o cursor Past EOL, keeps X position of cursor
18683:   eoNoCaret,                    //Makes it so the caret is never visible
18684:   eoNoSelection,                //Disables selecting text
18685:   eoRightMouseMovesCursor,      //When clicking with right mouse for popup menu, moves cursor to location
18686:   eoScrollByOneLess,            //Forces scrolling to be one less
18687:   eoScrollHintFollows,          //The scroll hint follows the mouse when scrolling vertically
18688:   eoScrollPastEof,              //Allows the cursor to go past the end of file marker
18689:   eoScrollPastEol,              //Allows cursor to go past last character into white space at end of a line
18690:   eoShowScrollHint,             //Shows a hint of the visible line numbers when scrolling vertically
18691:   eoShowSpecialChars,           //Shows the special Characters
18692:   eoSmartTabDelete,             //similar to Smart Tabs, but when you delete characters
18693:   eoSmartTabs,                  //When tabbing, cursor will go to non-white space character of previous line
18694:   eoSpecialLineDefaultFg,       //disables the foreground text color override using OnSpecialLineColor event
18695:   eoTabIndent,                  //If active <Tab>and<Shift><Tab> act block indent,unindent when text select
18696:   eoTabsToSpaces,               //Converts a tab character to a specified number of space characters
18697:   eoTrimTrailingSpaces          //Spaces at the end of lines will be trimmed and not saved
18698:
18699:   *****Important Editor Short Cuts*****);
18700: Double click to select a word and count words with highlightning.
18701: Triple click to select a line.
18702: CTRL+SHIFT+click to extend a selection.
18703: Drag with the ALT key down to select columns of text !!!
18704: Drag and drop is supported.
18705: Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
18706: Type CTRL+A to select all.
18707: Type CTRL+N to set a new line.
18708: Type CTRL+T to delete a line or token. //Tokenizer
18709: Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
18710: Type CTRL+Shift+T to add ToDo in line and list.
18711: Type CTRL+Shift+[0..9] to set bookmarks. //Bookmark
18712: Type CTRL[0..9] to jump or get to bookmarks.
18713: Type Home to position cursor at beginning of current line and End to position it at end of line.
18714: Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of document.
18715: Page Up and Page Down work as expected.
18716: CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
18717: using http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
18718:
18719: {$ Short Key Positions Ctrl<A-Z>: }
18720: def
18721:   <A> Select All
18722:   <B> Count Words
18723:   <C> Copy
18724:   <D> Internet Start
18725:   <E> Script List
18726:   <F> Find
18727:   <G> Goto
18728:   <H> Mark Line
18729:   <I> Interface List
18730:   <J> Code Completion
18731:   <K> Console
18732:   <L> Interface List Box
18733:   <M> Font Larger -
18734:   <N> New Line

```

```

18735: <O> Open File
18736: <P> Font Smaller +
18737: <Q> Quit
18738: <R> Replace
18739: <S> Save!
18740: <T> Delete Line
18741: <U> Use Case Editor
18742: <V> Paste
18743: <W> URI Links
18744: <X> Reserved for coding use internal
18745: <Y> Delete Line
18746: <Z> Undo
18747:
18748: ref
18749: F1 Help
18750: F2 Syntax Check
18751: F3 Search Next
18752: F4 New Instance
18753: F5 Line Mark /Breakpoint
18754: F6 Goto End
18755: F7 Debug Step Into
18756: F8 Debug Step Out
18757: F9 Compile
18758: F10 Menu
18759: F11 Word Count Highlight
18760: F12 Reserved for coding use internal
18761:
18762: def ReservedWords: array[0..82] of string =
18763:   ('and', 'array', 'as', 'asm', 'at', 'begin', 'case', 'class', 'const',
18764:    'constructor', 'default', 'destructor', 'dispinterface', 'div', 'do',
18765:    'downto', 'else', 'end', 'except', 'exports', 'file', 'finalization',
18766:    'finally', 'for', 'function', 'goto', 'if', 'implementation', 'in',
18767:    'inherited', 'initialization', 'inline', 'interface', 'is', 'label',
18768:    'library', 'message', 'mod', 'nil', 'not', 'object', 'of', 'on', 'or',
18769:    'out', 'packed', 'procedure', 'program', 'property', 'raise', 'read',
18770:    'record', 'repeat', 'resourcestring', 'set', 'shl', 'shr', 'string',
18771:    'stored', 'then', 'threadvar', 'to', 'try', 'type', 'unit', 'until',
18772:    'uses', 'var', 'while', 'with', 'write', 'xor', 'private', 'protected',
18773:    'public', 'published', 'def', 'ref', 'using', 'typedef', 'memol', 'memo2', 'doc', 'maxform1';
18774:   AllowedChars: array[0..5] of string = ('(', ')', '[', ']', ' ', 't', 'l', 't2', 't3: boolean;
18775:
18776: //-----
18777: //*****End of mX4 Public Tools API *****
18778: //-----
18779:
18780: Amount of Functions: 12130
18781: Amount of Procedures: 7566
18782: Amount of Constructors: 1240
18783: Totals of Calls: 20936
18784: SHA1: Win 3.9.9.94 780F1F749EFEB5C0FAA09C51B6AB048C6A00CC6
18785:
18786: *****
18787: Doc Short Manual with 50 Tips!
18788: *****
18789: - Install: just save your maxboxdef.ini before and then extract the zip file!
18790: - Toolbar: Click on the red maxbox Sign (right on top) opens your work directory or jump to <Help>
18791: - Menu: With <F2> you check syntax with <F8> you debug and <F9> you compile!
18792: - Menu: With <Ctrl><F3> you can search for code on examples
18793: - Menu: Open in menu Output a new instance <F4> of the box to compare or prepare your scripts
18794: - Menu: Set Interface Naviagator in menu /View/Intf Navigator
18795: - Menu: Switch or toogle between the last 2 scripts in menu File/LoadLast (History is set to 9 files)
18796:
18797: - Inifile: Set memory report in ini: MEMORYREPORT=Y :report on memory leaks on shutdown by dialog
18798: - Inifile: Refresh (reload) the inifile after edit with ../Help/Config Update
18799: - Context Menu: You can printout your scripts as a pdf-file or html-export
18800: - Context: You do have a context menu with the right mouse click
18801:
18802: - Menu: With the UseCase Editor you can convert graphic formats too.
18803: - Menu: On menu Options you find Addons as compiled scripts
18804: - IDE: You don't need a mouse to handle maxbox, use shortcuts
18805: - Menu: Check Options/ProcessMessages! if something is wrong or you can't see graphics in a time
18806: - IDE: Dragndrop your scripts in box or the model in use case editor (Cut,Copy,Paste always available)
18807: - Editor: You can get templates as code completion with <ctrl j> in editor like classp or iinterface
18808:   or ttimer (you type classp and then CTRL J),or you type tstringlist and <Ctrl><J>
18809:
18810: - Menu: In menu output (console) you can set output menu in edit mode by unchecking <read only output>
18811: - Editor: After the end. you can write or copy notes or descriptions concerning the app or code
18812: - Code: If you code a loop till key-pressed use function: isKeyPressed;
18813: - Code: Macro set the macros #name, #date, #host, #path, #file, #head #sign, Tutorial maxbox_starter25.pdf
18814: - Code: change Syntax in autoboot macro 'maxbootscript.txt'
18815: - Editor: - <F11> Click on Word in Editor search amount of words with highlighting, Dbl Click on Bookmarks
18816:   to delete and Click and mark to drag a bookmark
18817: - Menu: To start handling from CD-ROM (read only mode) uncheck in Menu /Options/Save before Compile
18818: - IDE: A file info with system and script information you find in menu Program/Information
18819: - IDE: After change the config file in help you can update changes in menu Help/Config Update
18820: - IDE: Make a screenshot of the content and environment in menu Output/Save Screenshot
18821: - IDE: Use a boot loader script 'maxbootscript.txt' (as auto start) to change box each time you start it.
18822: - IDE: With escape or <Ctrl> Q you can also leave the box or stop a script in menu program - stop program
18823: - Editor: Set Bookmarks to check your work in app or code

```

```

18824: - Editor: With <Ctrl H> you set {Active Line Color} and F11 you get Word Count Statistic on Output too
18825: - Editor: With {//TODO: some description} or DONE you set code entries for ToDo List in ../Help/ToDo List
18826: - Editor: With <Ctrl W> you set active URL links in your code to test availability in Context Menu
18827:
18828: - IDE with menu /Options/ADO SQL Workbench you can manage your Database
18829: - Context Menu: You can write your docs with RichEdit RTF printout /Editor Form Options/Richedit
18830: - Menu: Set Interface Navigator also with toggle <Ctrl L> or /View/Intf Navigator
18831: - Toolbar: In menu /View switch Off Toolbar and Coolbar to get more coding space
18832: - Code: Put some resources in your code /Help/Resource Explorer like bitbtn, forms, dialogs;
18833: - Code Editor: Compile with <F9> but also Alt C in case <F9> isn't available;
18834: - IDE set bookmarks with <Ctrl Shift> (0..9) and jump with <Ctrl> (0..9)
18835: - IDE menu /Help/Tools/ write with RTF Editor or open a DOS Shell or browse
18836:
18837: - Add on write your Requirements in RTF Docu with <CTRL ALT R> in context menu
18838: - Add on when no browser is available start /Options/Add ons/Easy Browser
18839: - Add on SOAP Tester with SOP POST File
18840: - Add on IP Protocol Sniffer with List View
18841: - Add on OpenGL mX Robot Demo for android
18842:
18843: - Menu: Help/Tools as a Tool Section with DOS Opener
18844: - Menu Editor: export the code as RTF File
18845: - Menu: Help/Tools/Syn Export your code as available in HTML or RTF
18846: - Menu: Help/Tools/ with <Alt H-S-D> you start the DOS Shell
18847: - Context: Auto Detect of Syntax depending on file extension
18848: - Code: some Windows API function start with w in the name like wGetAtomName();
18849: - IDE Close - if you can't close the box then reset it <Ctrl F2> in menu /Debug
18850: - IDE File Check with menu ../View/File Changes/...
18851:
18852: - using DLL example in maXbox: //function: {*****}
18853:   Function GetProcessMemoryInfo(Process: THandle; var MemoryCounters: TProcessMemoryCounters;
18854:     cb: DWORD): BOOL; //stdcall;
18855:   External 'GetProcessMemoryInfo@psapi.dll stdcall';
18856:   Function OpenProcess(dwDesiredAccess: DWORD; bInheritHandle: BOOL; dwProcessId: DWORD): THandle;
18857:   External 'OpenProcess@kernel32.dll stdcall';
18858:
18859: PCT Precompile Technology , mX4 ScriptStudio
18860: Indy, JCL, Jedi, VCL, Systools, TurboPower, Fundamentals, ExtendedRTL, Synedit
18861: DMath, devC, Graphics32, ExtPascal, mX4, LCL, CLX, FCL, CPort and more
18862: emacs layers: system-package-component-unit-class-function-block
18863: new keywords def ref using maXCalcF
18864: UML: use case act class state seq pac comp dep - lib lab
18865: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
18866: Tutorials, 30 Units add, VCL constructors, controls plus, unit list
18867: 2 Tutorials, 36 Units add, Synapse V40, LDAP, OpenSSL, AVScan
18868: 1 Report, 15 Units add, DBCtrls, Stream+, IRadio, Wininet
18869: 1 DLL Report, 19 Units add, DRTTable, Remote+, Cindy functions!
18870: https://unibe-ch.academia.edu/MaxKleiner
18871: www.slideshare.net/maxkleiner1
18872: http://www.scribd.com/max\_kleiner
18873:
18874:
18875:
18876:
18877:
18878: *****
18879: unit List asm internal end
18880: *****
18881: 01 unit RRegister_StrUtils_Routines(exec); //Delphi
18882: 02 unit SRegister_IdStrings //Indy Sockets
18883: 03 unit RRegister_niSTRING_Routines(exec); //from RegEx
18884: 04 unit uPSI_fMain Functions; //maXbox Open Tools API
18885: 05 unit IFSI_WinFormlpuzzle; //maXbox
18886: 06 unit RRegister_LinarBitmap_Routines(exec); //ImageFileLibBCB
18887: 07 unit RegisterDateTimeLibrary_R(exec); //Delphi
18888: 08 unit RRegister_MathMax_Routines(exec); //Jedi & Delphi
18889: 09 unit RRegister_IdGlobal_Routines(exec); //Indy Sockets
18890: 10 unit RRegister_SysUtils_Routines(exec); //Delphi
18891: 11 unit uPSI_IdTCPConnection; //Indy some functions
18892: 12 unit uPSCCompiler.pas; //PS kernel functions
18893: 13 unit uPSI_DBCommon; //DB Common_Routines and Types
18894: 14 unit uPSI_Printers.pas //Delphi VCL
18895: 15 unit uPSI_MPlayer.pas //Delphi VCL
18896: 16 unit uPSCComobj; //COM Functions
18897: 17 unit uPSI_Clipbrd; //Delphi VCL
18898: 18 unit Filectrl in IFSI_SysUtils_max; //VCL Runtime
18899: 19 unit uPSI_SqlExpr; //DEX3
18900: 20 unit uPSI_ADODB; //ADODB
18901: 21 unit uPSI_StrHlpr; //String Helper Routines
18902: 22 unit uPSI_DateUtils; //Expansion to DateTimeLib
18903: 23 unit uPSI_FileUtils; //Expansion to Sys/File Utils
18904: 24 unit JUtils / gsUtils; //Jedi / Metabase
18905: 25 unit JvFunctions_max; //Jedi Functions
18906: 26 unit HTTPParser; //Delphi VCL
18907: 27 unit HTTPUtil; //Delphi VCL
18908: 28 unit uPSI_XMLUtil; //Delphi VCL
18909: 29 unit uPSI_SOAPHTTPClient; //Delphi VCL SOAP Webservice V3.5
18910: 30 unit uPSI_Contrns; //Delphi RTL Container of Classes
18911: 31 unit uPSI_MaskUtils; //RTL Edit and Mask functions
18912: 32 unit uPSI_MyBigInt; //big integer class with Math

```

```

18913: 33 unit uPSI_ConvUtils; //Delphi VCL Conversions engine
18914: 34 unit Types_Variants; //Delphi\Win32\rtl\sys
18915: 35 unit uPSI_IdHashSHA1; //Indy Crypto Lib
18916: 36 unit uPSI_IdHashMessageDigest //Indy Crypto;
18917: 37 unit uPSI_IdASN1Util; //Indy ASN1Utility Routines;
18918: 38 unit uPSI_IdLogFile; //Indy Logger from LogBase
18919: 39 unit uPSI_IdIcmpClient; //Indy Ping ICMP
18920: 40 unit uPSI_IdHashMessageDigest_max //Indy Crypto &OpenSSL;
18921: 41 unit uPSI_FileCtrl; //Delphi RTL
18922: 42 unit uPSI_Outline; //Delphi VCL
18923: 43 unit uPSI_ScktComp; //Delphi RTL
18924: 44 unit uPSI_Calendar; //Delphi VCL
18925: 45 unit uPSI_VListView; //VListView;
18926: 46 unit uPSI_DBGrids; //Delphi VCL
18927: 47 unit uPSI_DBCtrls; //Delphi VCL
18928: 48 unit ide_debugoutput; //maXbox
18929: 49 unit uPSI_ComCtrls; //Delphi VCL
18930: 50 unit uPSC_stdCtrls; //Delphi VCL
18931: 51 unit uPSI_Dialogs; //Delphi VCL
18932: 52 unit uPSI_StdConvs; //Delphi RTL
18933: 53 unit uPSI_DBClient; //Delphi RTL
18934: 54 unit uPSI_DBPlatform; //Delphi RTL
18935: 55 unit uPSI_Provider; //Delphi RTL
18936: 56 unit uPSI_FMTBcd; //Delphi RTL
18937: 57 unit uPSI_DBCGrids; //Delphi VCL
18938: 58 unit uPSI_CDSUtil; //MIDAS
18939: 59 unit uPSI_VarHlpr; //Delphi RTL
18940: 60 unit uPSI_ExtDlgs; //Delphi VCL
18941: 61 unit sdpStopwatch; //maXbox
18942: 62 unit uPSI_JclStatistics; //JCL
18943: 63 unit uPSI_JclLogic; //JCL
18944: 64 unit uPSI_JclMiscel; //JCL
18945: 65 unit uPSI_JclMath_max; //JCL RTL
18946: 66 unit uPSI_uTPLb_StreamUtils; //LockBox 3
18947: 67 unit uPSI_MathUtils; //BCB
18948: 68 unit uPSI_JclMultimedia; //JCL
18949: 69 unit uPSI_WideStrUtils; //Delphi API/RTL
18950: 70 unit uPSI_GraphUtil; //Delphi RTL
18951: 71 unit uPSI_TypeTrans; //Delphi RTL
18952: 72 unit uPSI_HTTPApp; //Delphi VCL
18953: 73 unit uPSI_DBWeb; //Delphi VCL
18954: 74 unit uPSI_DBBdeWeb; //Delphi VCL
18955: 75 unit uPSI_DBXpressWeb; //Delphi VCL
18956: 76 unit uPSI_ShadowWnd; //Delphi VCL
18957: 77 unit uPSI_ToolWin; //Delphi VCL
18958: 78 unit uPSI_Tabs; //Delphi VCL
18959: 79 unit uPSI_JclGraphUtils; //JCL
18960: 80 unit uPSI_JclCounter; //JCL
18961: 81 unit uPSI_JclSysInfo; //JCL
18962: 82 unit uPSI_JclSecurity; //JCL
18963: 83 unit uPSI_JclFileUtils; //JCL
18964: 84 unit uPSI_IdUserAccounts; //Indy
18965: 85 unit uPSI_IdAuthentication; //Indy
18966: 86 unit uPSI_uTPLb_AES; //LockBox 3
18967: 87 unit uPSI_IdHashSHA1; //LockBox 3
18968: 88 unit uTPLb_BlockCipher; //LockBox 3
18969: 89 unit uPSI_ValEdit.pas; //Delphi VCL
18970: 90 unit uPSI_JvVCLUtils; //JCL
18971: 91 unit uPSI_JvDBUtil; //JCL
18972: 92 unit uPSI_JvDBUtils; //JCL
18973: 93 unit uPSI_JvAppUtils; //JCL
18974: 94 unit uPSI_JvCtrlUtils; //JCL
18975: 95 unit uPSI_JvFormToHtml; //JCL
18976: 96 unit uPSI_JvParsing; //JCL
18977: 97 unit uPSI_SerDlgs; //Toolbox
18978: 98 unit uPSI_Serial; //Toolbox
18979: 99 unit uPSI_JvComponent; //JCL
18980: 100 unit uPSI_JvCalc; //JCL
18981: 101 unit uPSI_JvBdeUtils; //JCL
18982: 102 unit uPSI_JvDateUtil; //JCL
18983: 103 unit uPSI_JvGenetic; //JCL
18984: 104 unit uPSI_JclBase; //JCL
18985: 105 unit uPSI_JvUtils; //JCL
18986: 106 unit uPSI_JvStrUtil; //JCL
18987: 107 unit uPSI_JvStrUtils; //JCL
18988: 108 unit uPSI_JvFileUtil; //JCL
18989: 109 unit uPSI_JvMemoryInfos; //JCL
18990: 110 unit uPSI_JvComputerInfo; //JCL
18991: 111 unit uPSI_JvgCommClasses; //JCL
18992: 112 unit uPSI_JvgLogics; //JCL
18993: 113 unit uPSI_JvLED; //JCL
18994: 114 unit uPSI_JvTurtle; //JCL
18995: 115 unit uPSI_SortThds; unit uPSI_ThSort; //maXbox
18996: 116 unit uPSI_JvgUtils; //JCL
18997: 117 unit uPSI_JvExprParser; //JCL
18998: 118 unit uPSI_HexDump; //Borland
18999: 119 unit uPSI_DBLogDlg; //VCL
19000: 120 unit uPSI_SqlTimSt; //RTL
19001: 121 unit uPSI_JvHtmlParser; //JCL

```



```

19002: 122 unit uPSI_JvgXMLSerializer; //JCL
19003: 123 unit uPSI_JvJCLUtils; //JCL
19004: 124 unit uPSI_JvStrings; //JCL
19005: 125 unit uPSI_uTPLb_IntegerUtils; //TurboPower
19006: 126 unit uPSI_uTPLb_HugeCardinal; //TurboPower
19007: 127 unit uPSI_uTPLb_HugeCardinalUtils; //TurboPower
19008: 128 unit uPSI_SynRegExpr; //SynEdit
19009: 129 unit uPSI_StUtils; //SysTools4
19010: 130 unit uPSI_StToHTML; //SysTools4
19011: 131 unit uPSI_StStrms; //SysTools4
19012: 132 unit uPSI_StFIN; //SysTools4
19013: 133 unit uPSI_StAstroP; //SysTools4
19014: 134 unit uPSI_StStat; //SysTools4
19015: 135 unit uPSI_StNetCon; //SysTools4
19016: 136 unit uPSI_StDecMth; //SysTools4
19017: 137 unit uPSI_StOStr; //SysTools4
19018: 138 unit uPSI_StPtrns; //SysTools4
19019: 139 unit uPSI_StNetMessage; //SysTools4
19020: 140 unit uPSI_StMath; //SysTools4
19021: 141 unit uPSI_StExpEng; //SysTools4
19022: 142 unit uPSI_StCRC; //SysTools4
19023: 143 unit uPSI_StExport; //SysTools4
19024: 144 unit uPSI_StExpLog; //SysTools4
19025: 145 unit uPSI_ActnList; //Delphi VCL
19026: 146 unit uPSI_jpeg; //Borland
19027: 147 unit uPSI_StRandom; //SysTools4
19028: 148 unit uPSI_StDict; //SysTools4
19029: 149 unit uPSI_StBCD; //SysTools4
19030: 150 unit uPSI_StTxtDat; //SysTools4
19031: 151 unit uPSI_StRegEx; //SysTools4
19032: 152 unit uPSI_IMouse; //VCL
19033: 153 unit uPSI_SyncObjs; //VCL
19034: 154 unit uPSI_AsyncCalls; //Hausladen
19035: 155 unit uPSI_ParallelJobs; //Saraiva
19036: 156 unit uPSI_Variants; //VCL
19037: 157 unit uPSI_VarCmplx; //VCL Wolfram
19038: 158 unit uPSI_DTDSchema; //VCL
19039: 159 unit uPSI_ShLwApi; //Brakel
19040: 160 unit uPSI_IBUtils; //VCL
19041: 161 unit uPSI_CheckLst; //VCL
19042: 162 unit uPSI_JvSimpleXml; //JCL
19043: 163 unit uPSI_JclSimpleXml; //JCL
19044: 164 unit uPSI_JvXmlDatabase; //JCL
19045: 165 unit uPSI_JvMaxPixel; //JCL
19046: 166 unit uPSI_JvItemsSearchs; //JCL
19047: 167 unit uPSI_StExpEng2; //SysTools4
19048: 168 unit uPSI_StGenLog; //SysTools4
19049: 169 unit uPSI_JvLogFile; //Jcl
19050: 170 unit uPSI_CPort; //ComPort Lib v4.11
19051: 171 unit uPSI_CPortCtl; //ComPort
19052: 172 unit uPSI_CPortEsc; //ComPort
19053: 173 unit BarCodeScanner; //ComPort
19054: 174 unit uPSI_JvGraph; //JCL
19055: 175 unit uPSI_JvComCtrls; //JCL
19056: 176 unit uPSI_GUITesting; //D Unit
19057: 177 unit uPSI_JvFindFiles; //JCL
19058: 178 unit uPSI_StSystem; //SysTools4
19059: 179 unit uPSI_JvKeyboardStates; //JCL
19060: 180 unit uPSI_JvMail; //JCL
19061: 181 unit uPSI_JclConsole; //JCL
19062: 182 unit uPSI_JclLANMan; //JCL
19063: 183 unit uPSI_IdCustomHTTPServer; //Indy
19064: 184 unit IdHTTPServer; //Indy
19065: 185 unit uPSI_IdTCPServer; //Indy
19066: 186 unit uPSI_IdSocketHandle; //Indy
19067: 187 unit uPSI_IdIOHandlerSocket; //Indy
19068: 188 unit IdIOHandler; //Indy
19069: 189 unit uPSI_cutils; //Bloodshed
19070: 190 unit uPSI_BoldUtils; //boldsoft
19071: 191 unit uPSI_IdSimpleServer; //Indy
19072: 192 unit uPSI_IdSSLOpenSSL; //Indy
19073: 193 unit uPSI_IdMultipartFormData; //Indy
19074: 194 unit uPSI_SynURIOpener; //SynEdit
19075: 195 unit uPSI_PperlRegEx; //PCRE
19076: 196 unit uPSI_IdHeaderList; //Indy
19077: 197 unit uPSI_StFirst; //SysTools4
19078: 198 unit uPSI_JvCtrls; //JCL
19079: 199 unit uPSI_IdTrivialFTPBase; //Indy
19080: 200 unit uPSI_IdTrivialFTP; //Indy
19081: 201 unit uPSI_IdUDPBase; //Indy
19082: 202 unit uPSI_IdUDPCClient; //Indy
19083: 203 unit uPSI_utypes; //for DMath.DLL
19084: 204 unit uPSI_ShellAPI; //Borland
19085: 205 unit uPSI_IdRemoteCMDClient; //Indy
19086: 206 unit uPSI_IdRemoteCMDServer; //Indy
19087: 207 unit IdRexecServer; //Indy
19088: 208 unit IdRexec; (unit uPSI_IdRexec;) //Indy
19089: 209 unit IdUDPServer; //Indy
19090: 210 unit IdTimeUDPServer; //Indy

```

```

19091: 211 unit IdTimeServer; //Indy
19092: 212 unit IdTimeUDP; (unit uPSI_IdUDPServer;) //Indy
19093: 213 unit uPSI_IdIPWatch; //Indy
19094: 214 unit uPSI_IdIrcServer; //Indy
19095: 215 unit uPSI_IdMessageCollection; //Indy
19096: 216 unit uPSI_cPEM; //Fundamentals 4
19097: 217 unit uPSI_cFundamentUtils; //Fundamentals 4
19098: 218 unit uPSI_uwinplot; //DMath
19099: 219 unit uPSI_xrtl_util_CPUUtils; //ExtendedRTL
19100: 220 unit uPSI_GR32_System; //Graphics32
19101: 221 unit uPSI_cFileUtils; //Fundamentals 4
19102: 222 unit uPSI_cDateTime; (timemachine) //Fundamentals 4
19103: 223 unit uPSI_cTimers; (high precision timer) //Fundamentals 4
19104: 224 unit uPSI_cRandom; //Fundamentals 4
19105: 225 unit uPSI_ueval; //DMath
19106: 226 unit uPSI_xrtl_net_URIUtils; //ExtendedRTL
19107: 227 unit xrtl_net_URIUtils; //ExtendedRTL
19108: 228 unit uPSI_ufft; (FFT) //DMath
19109: 229 unit uPSI_DBXChannel; //Delphi
19110: 230 unit uPSI_DBXIndyChannel; //Delphi Indy
19111: 231 unit uPSI_xrtl_util_COMCat; //ExtendedRTL
19112: 232 unit uPSI_xrtl_util_StrUtils; //ExtendedRTL
19113: 233 unit uPSI_xrtl_util_VariantUtils; //ExtendedRTL
19114: 234 unit uPSI_xrtl_util_FileUtils; //ExtendedRTL
19115: 235 unit xrtl_util_Compat; //ExtendedRTL
19116: 236 unit uPSI_OleAuto; //Borland
19117: 237 unit uPSI_xrtl_util_COMUtils; //ExtendedRTL
19118: 238 unit uPSI_CmAdmCtl; //Borland
19119: 239 unit uPSI_ValEdit2; //VCL
19120: 240 unit uPSI_GR32; //Graphics32
19121: 241 unit uPSI_GR32_Image; //Graphics32
19122: 242 unit uPSI_xrtl_util_TimeUtils; //ExtendedRTL
19123: 243 unit uPSI_xrtl_util_TimeZone; //ExtendedRTL
19124: 244 unit uPSI_xrtl_util_TimeStamp; //ExtendedRTL
19125: 245 unit uPSI_xrtl_util_Map; //ExtendedRTL
19126: 246 unit uPSI_xrtl_util_Set; //ExtendedRTL
19127: 247 unit uPSI_CPortMonitor; //ComPort
19128: 248 unit uPSI_StIniStm; //SysTools4
19129: 249 unit uPSI_GR32_ExtImage; //Graphics32
19130: 250 unit uPSI_GR32_OrdinalMaps; //Graphics32
19131: 251 unit uPSI_GR32_Rasterizers; //Graphics32
19132: 252 unit uPSI_xrtl_util_Exception; //ExtendedRTL
19133: 253 unit uPSI_xrtl_util_Value; //ExtendedRTL
19134: 254 unit uPSI_xrtl_util_Compare; //ExtendedRTL
19135: 255 unit uPSI_FlatSB; //VCL
19136: 256 unit uPSI_JvAnalogClock; //JCL
19137: 257 unit uPSI_JvAlarms; //JCL
19138: 258 unit uPSI_JvSQLS; //JCL
19139: 259 unit uPSI_JvDBSecur; //JCL
19140: 260 unit uPSI_JvDBQBE; //JCL
19141: 261 unit uPSI_JvStarfield; //JCL
19142: 262 unit uPSI_JVCLMiscal; //JCL
19143: 263 unit uPSI_JvProfiler32; //JCL
19144: 264 unit uPSI_JvDirectories; //JCL
19145: 265 unit uPSI_JclSchedule; //JCL
19146: 266 unit uPSI_JclSvcCtrl; //JCL
19147: 267 unit uPSI_JvSoundControl; //JCL
19148: 268 unit uPSI_JvBDESQLScript; //JCL
19149: 269 unit uPSI_JvgDigits; //JCL>
19150: 270 unit uPSI_ImgList; //TCustomImageList
19151: 271 unit uPSI_JclMIDI; //JCL>
19152: 272 unit uPSI_JclWinMidi; //JCL>
19153: 273 unit uPSI_JclNTFS; //JCL>
19154: 274 unit uPSI_JclAppInst; //JCL>
19155: 275 unit uPSI_JvRle; //JCL>
19156: 276 unit uPSI_JvRas32; //JCL>
19157: 277 unit uPSI_JvImageDrawThread; //JCL>
19158: 278 unit uPSI_JvImageWindow; //JCL>
19159: 279 unit uPSI_JvTransparentForm; //JCL>
19160: 280 unit uPSI_JvWinDialogs; //JCL>
19161: 281 unit uPSI_JvSimLogic; //JCL>
19162: 282 unit uPSI_JvSimIndicator; //JCL>
19163: 283 unit uPSI_JvSimPID; //JCL>
19164: 284 unit uPSI_JvSimPIDLinker; //JCL>
19165: 285 unit uPSI_IdRFCReply; //Indy
19166: 286 unit uPSI_IdIdent; //Indy
19167: 287 unit uPSI_IdIdentServer; //Indy
19168: 288 unit uPSI_JvPatchFile; //JCL
19169: 289 unit uPSI_StNetPfm; //SysTools4
19170: 290 unit uPSI_StNet; //SysTools4
19171: 291 unit uPSI_JclPeImage; //JCL
19172: 292 unit uPSI_JclPrint; //JCL
19173: 293 unit uPSI_JclMime; //JCL
19174: 294 unit uPSI_JvRichEdit; //JCL
19175: 295 unit uPSI_JvDBRichEd; //JCL
19176: 296 unit uPSI_JvDice; //JCL
19177: 297 unit uPSI_JvFloatEdit; //JCL 3.9.8
19178: 298 unit uPSI_JvDirFrm; //JCL
19179: 299 unit uPSI_JvDualList; //JCL

```

```

19180: 300 unit uPSI_JvSwitch; //JCL
19181: 301 unit uPSI_JvTimerLst; //JCL
19182: 302 unit uPSI_JvMemTable; //JCL
19183: 303 unit uPSI_JvObjStr; //JCL
19184: 304 unit uPSI_StLArr; //SysTools4
19185: 305 unit uPSI_StWmDCpy; //SysTools4
19186: 306 unit uPSI_StText; //SysTools4
19187: 307 unit uPSI_StNTLog; //SysTools4
19188: 308 unit uPSI_xrtl_math_Integer; //ExtendedRTL
19189: 309 unit uPSI_JvImagPrvw; //JCL
19190: 310 unit uPSI_JvFormPatch; //JCL
19191: 311 unit uPSI_JvPicClip; //JCL
19192: 312 unit uPSI_JvDataConv; //JCL
19193: 313 unit uPSI_JvCpuUsage; //JCL
19194: 314 unit uPSI_JclUnitConv_mX2; //JCL
19195: 315 unit JvDualListForm; //JCL
19196: 316 unit uPSI_JvCpuUsage2; //JCL
19197: 317 unit uPSI_JvParserForm; //JCL
19198: 318 unit uPSI_JvJanTreeView; //JCL
19199: 319 unit uPSI_JvTransLED; //JCL
19200: 320 unit uPSI_JvPlaylist; //JCL
19201: 321 unit uPSI_JvFormAutoSize; //JCL
19202: 322 unit uPSI_JvYearGridEditForm; //JCL
19203: 323 unit uPSI_JvMarkupCommon; //JCL
19204: 324 unit uPSI_JvChart; //JCL
19205: 325 unit uPSI_JvXPCore; //JCL
19206: 326 unit uPSI_JvXPCoreUtils; //JCL
19207: 327 unit uPSI_StatsClasses; //mX4
19208: 328 unit uPSI_ExtCtrls2; //VCL
19209: 329 unit uPSI_JvUrlGrabbers; //JCL
19210: 330 unit uPSI_JvXmlTree; //JCL
19211: 331 unit uPSI_JvWavePlayer; //JCL
19212: 332 unit uPSI_JvUnicodeCanvas; //JCL
19213: 333 unit uPSI_JvTFUtils; //JCL
19214: 334 unit uPSI_IdServerIOHandler; //Indy
19215: 335 unit uPSI_IdServerIOHandlerSocket; //Indy
19216: 336 unit uPSI_IdMessageCoder; //Indy
19217: 337 unit uPSI_IdMessageCoderMIME; //Indy
19218: 338 unit uPSI_IdMIMETypes; //Indy
19219: 339 unit uPSI_JvConverter; //JCL
19220: 340 unit uPSI_JvCsvParse; //JCL
19221: 341 unit uPSI_umath; unit uPSI_ugamma; //DMath
19222: 342 unit uPSI_ExcelExport; (Nat:TJsExcelExport) //JCL
19223: 343 unit uPSI_JvDBGridExport; //JCL
19224: 344 unit uPSI_JvgExport; //JCL
19225: 345 unit uPSI_JvSerialMaker; //JCL
19226: 346 unit uPSI_JvWin32; //JCL
19227: 347 unit uPSI_JvPaintFX; //JCL
19228: 348 unit uPSI_JvOracleDataSet; (beta) //JCL
19229: 349 unit uPSI_JvValidators; (preview) //JCL
19230: 350 unit uPSI_JvNTEventLog; //JCL
19231: 351 unit uPSI_ShellZipTool; //mX4
19232: 352 unit uPSI_JvJoystick; //JCL
19233: 353 unit uPSI_JvMailSlots; //JCL
19234: 354 unit uPSI_JclComplex; //JCL
19235: 355 unit uPSI_SynPdf; //Synopse
19236: 356 unit uPSI_Registry; //VCL
19237: 357 unit uPSI_TlHelp32; //VCL
19238: 358 unit uPSI_JclRegistry; //JCL
19239: 359 unit uPSI_JvAirBrush; //JCL
19240: 360 unit uPSI_mORMotReport; //Synopse
19241: 361 unit uPSI_JclLocales; //JCL
19242: 362 unit uPSI_SynEdit; //SynEdit
19243: 363 unit uPSI_SynEditTypes; //SynEdit
19244: 364 unit uPSI_SynMacroRecorder; //SynEdit
19245: 365 unit uPSI_LongIntList; //SynEdit
19246: 366 unit uPSI_devcutils; //DevC
19247: 367 unit uPSI_SynEditMiscClasses; //SynEdit
19248: 368 unit uPSI_SynEditRegexSearch; //SynEdit
19249: 369 unit uPSI_SynEditHighlighter; //SynEdit
19250: 370 unit uPSI_SynHighlighterPas; //SynEdit
19251: 371 unit uPSI_JvSearchFiles; //JCL
19252: 372 unit uPSI_SynHighlighterAny; //Lazarus
19253: 373 unit uPSI_SynEditKeyCmds; //SynEdit
19254: 374 unit uPSI_SynEditMiscProcs; //SynEdit
19255: 375 unit uPSI_SynEditKbdHandler; //SynEdit
19256: 376 unit uPSI_JvAppInst; //JCL
19257: 377 unit uPSI_JvAppEvent; //JCL
19258: 378 unit uPSI_JvAppCommand; //JCL
19259: 379 unit uPSI_JvAnimTitle; //JCL
19260: 380 unit uPSI_JvAnimatedImage; //JCL
19261: 381 unit uPSI_SynEditExport; //SynEdit
19262: 382 unit uPSI_SynExportHTML; //SynEdit
19263: 383 unit uPSI_SynExportRTF; //SynEdit
19264: 384 unit uPSI_SynEditSearch; //SynEdit
19265: 385 unit uPSI_fMain_back; //maXbox;
19266: 386 unit uPSI_JvZoom; //JCL
19267: 387 unit uPSI_PMrand; //PM
19268: 388 unit uPSI_JvSticker; //JCL

```

```

19269: 389 unit uPSI_XmlVerySimple; //mX4
19270: 390 unit uPSI_Services; //ExtPascal
19271: 391 unit uPSI_ExtPascalUtils; //ExtPascal
19272: 392 unit uPSI_SocketsDelphi; //ExtPascal
19273: 393 unit uPSI_StBarC; //SysTools
19274: 394 unit uPSI_StDbBarC; //SysTools
19275: 395 unit uPSI_StBarPN; //SysTools
19276: 396 unit uPSI_StDbPNBC; //SysTools
19277: 397 unit uPSI_StDb2DBC; //SysTools
19278: 398 unit uPSI_StMoney; //SysTools
19279: 399 unit uPSI_JvForth; //JCL
19280: 400 unit uPSI_RestRequest; //mX4
19281: 401 unit uPSI_HttpRESTConnectionIndy; //mX4
19282: 402 unit uPSI_JvXmlDatabase; //update //JCL
19283: 403 unit uPSI_StAstro; //SysTools
19284: 404 unit uPSI_StSort; //SysTools
19285: 405 unit uPSI_StDate; //SysTools
19286: 406 unit uPSI_StDateSt; //SysTools
19287: 407 unit uPSI_StBase; //SysTools
19288: 408 unit uPSI_StVInfo; //SysTools
19289: 409 unit uPSI_JvBrowseFolder; //JCL
19290: 410 unit uPSI_JvBoxProcs; //JCL
19291: 411 unit uPSI_urandom; (unit uranuvag;) //DMath
19292: 412 unit uPSI_usimann; (unit ugenalg;) //DMath
19293: 413 unit uPSI_JvHighlighter; //JCL
19294: 414 unit uPSI_Diff; //mX4
19295: 415 unit uPSI_SpringWinAPI; //DSpring
19296: 416 unit uPSI_StBits; //SysTools
19297: 417 unit uPSI_TomDBQueue; //mX4
19298: 418 unit uPSI_MultilangTranslator; //mX4
19299: 419 unit uPSI_HyperLabel; //mX4
19300: 420 unit uPSI_Starter; //mX4
19301: 421 unit uPSI_FileAssocs; //devC
19302: 422 unit uPSI_devFileMonitorX; //devC
19303: 423 unit uPSI_devrun; //devC
19304: 424 unit uPSI_devExec; //devC
19305: 425 unit uPSI_oysUtils; //devC
19306: 426 unit uPSI_DosCommand; //devC
19307: 427 unit uPSI_CppTokenizer; //devC
19308: 428 unit uPSI_JvHLPParser; //devC
19309: 429 unit uPSI_JclMapi; //JCL
19310: 430 unit uPSI_JclShell; //JCL
19311: 431 unit uPSI_JclCOM; //JCL
19312: 432 unit uPSI_GR32_Math; //Graphics32
19313: 433 unit uPSI_GR32_LowLevel; //Graphics32
19314: 434 unit uPSI_SimpleHl; //mX4
19315: 435 unit uPSI_GR32_Filters; //Graphics32
19316: 436 unit uPSI_GR32_VectorMaps; //Graphics32
19317: 437 unit uPSI_oXMLFunctions; //Fundamentals 4
19318: 438 unit uPSI_JvTimer; //JCL
19319: 439 unit uPSI_cHTTPUtils; //Fundamentals 4
19320: 440 unit uPSI_cTLSUtils; //Fundamentals 4
19321: 441 unit uPSI_JclGraphics; //JCL
19322: 442 unit uPSI_JclSynch; //JCL
19323: 443 unit uPSI_IdTelnet; //Indy
19324: 444 unit uPSI_IdTelnetServer; //Indy
19325: 445 unit uPSI_IdEcho; //Indy
19326: 446 unit uPSI_IdEchoServer; //Indy
19327: 447 unit uPSI_IdEchoUDP; //Indy
19328: 448 unit uPSI_IdEchoUDPServer; //Indy
19329: 449 unit uPSI_IdSocks; //Indy
19330: 450 unit uPSI_IdAntiFreezeBase; //Indy
19331: 451 unit uPSI_IdHostnameServer; //Indy
19332: 452 unit uPSI_IdTunnelCommon; //Indy
19333: 453 unit uPSI_IdTunnelMaster; //Indy
19334: 454 unit uPSI_IdTunnelSlave; //Indy
19335: 455 unit uPSI_IdRSH; //Indy
19336: 456 unit uPSI_IdRSHServer; //Indy
19337: 457 unit uPSI_Spring_Cryptography_Utils; //Spring4Delphi
19338: 458 unit uPSI_MapReader; //devC
19339: 459 unit uPSI_LibTar; //devC
19340: 460 unit uPSI_IdStack; //Indy
19341: 461 unit uPSI_IdBlockCipherIntercept; //Indy
19342: 462 unit uPSI_IdChargenServer; //Indy
19343: 463 unit uPSI_IdFTPServer; //Indy
19344: 464 unit uPSI_IdException; //Indy
19345: 465 unit uPSI_utexplot; //DMath
19346: 466 unit uPSI_uwinstr; //DMath
19347: 467 unit uPSI_VarRecUtils; //devC
19348: 468 unit uPSI_JvStringListToHtml; //JCL
19349: 469 unit uPSI_JvStringHolder; //JCL
19350: 470 unit uPSI_IdCoder; //Indy
19351: 471 unit uPSI_SynHighlighterDfm; //Synedit
19352: 472 unit uHighlighterProcs; in 471 //Synedit
19353: 473 unit uPSI_LazFileUtils; //LCL
19354: 474 unit uPSI_IDECmdLine; //LCL
19355: 475 unit uPSI_lazMasks; //LCL
19356: 476 unit uPSI_ip_misc; //mX4
19357: 477 unit uPSI_Barcode; //LCL

```



```

19358: 478 unit uPSI_SimpleXML; //LCL
19359: 479 unit uPSI_JclIniFiles; //JCL
19360: 480 unit uPSI_D2XXUnit; {$X-} //FTDI
19361: 481 unit uPSI_JclDateTime; //JCL
19362: 482 unit uPSI_JclEDI; //JCL
19363: 483 unit uPSI_JclMiscel2; //JCL
19364: 484 unit uPSI_JclValidation; //JCL
19365: 485 unit uPSI_JclAnsiStrings; {-PString} //JCL
19366: 486 unit uPSI_SynEditMiscProcs2; //Synedit
19367: 487 unit uPSI_JclStreams; //JCL
19368: 488 unit uPSI_QRCode; //mX4
19369: 489 unit uPSI_BlockSocket; //ExtPascal
19370: 490 unit uPSI_Masks,Utills //VCL
19371: 491 unit uPSI_synauttil; //Synapse!
19372: 492 unit uPSI_JclMath_Class; //JCL RTL
19373: 493 unit ugamdist; //Gamma function //DMath
19374: 494 unit uibeta, ucorrel; //IBeta //DMath
19375: 495 unit uPSI_SRMgr; //mX4
19376: 496 unit uPSI_HotLog; //mX4
19377: 497 unit uPSI_DebugBox; //mX4
19378: 498 unit uPSI_ustrings; //DMath
19379: 499 unit uPSI_uregtest; //DMath
19380: 500 unit uPSI_usimplex; //DMath
19381: 501 unit uPSI_uhyper; //DMath
19382: 502 unit uPSI_IdHLL7; //Indy
19383: 503 unit uPSI_IdIPMCastBase; //Indy
19384: 504 unit uPSI_IdIPMCastServer; //Indy
19385: 505 unit uPSI_IdIPMCastClient; //Indy
19386: 506 unit uPSI_unlfit; //nlregression //DMath
19387: 507 unit uPSI_IdRawHeaders; //Indy
19388: 508 unit uPSI_IdRawClient; //Indy
19389: 509 unit uPSI_IdRawFunctions; //Indy
19390: 510 unit uPSI_IdTCPStream; //Indy
19391: 511 unit uPSI_IdSNPP; //Indy
19392: 512 unit uPSI_St2DBarC; //SysTools
19393: 513 unit uPSI_ImageWin; //FTL //VCL
19394: 514 unit uPSI_CustomDrawTreeView; //FTL //VCL
19395: 515 unit uPSI_GraphWin; //FTL //VCL
19396: 516 unit uPSI_actionMain; //FTL //VCL
19397: 517 unit uPSI_StSpawn; //SysTools
19398: 518 unit uPSI_CtlPanel; //VCL
19399: 519 unit uPSI_IdLPR; //Indy
19400: 520 unit uPSI_SockRequestInterpreter; //Indy
19401: 521 unit uPSI_ulambert; //DMath
19402: 522 unit uPSI_ucholesk; //DMath
19403: 523 unit uPSI_SimpleDS; //VCL
19404: 524 unit uPSI_DBXSqlScanner; //VCL
19405: 525 unit uPSI_DBXMetaDataUtil; //VCL
19406: 526 unit uPSI_Chart; //TEE
19407: 527 unit uPSI_TeeProcs; //TEE
19408: 528 unit mXBDEUtils; //mX4
19409: 529 unit uPSI_MDIEdit; //VCL
19410: 530 unit uPSI_CopyPrsr; //VCL
19411: 531 unit uPSI_SockApp; //VCL
19412: 532 unit uPSI_AppEvtnts; //VCL
19413: 533 unit uPSI_ExtActns; //VCL
19414: 534 unit uPSI_TeEngine; //TEE
19415: 535 unit uPSI_CoolMain; //browser //VCL
19416: 536 unit uPSI_StCRC; //SysTools
19417: 537 unit uPSI_StDecMth2; //SysTools
19418: 538 unit uPSI_frmExportMain; //Synedit
19419: 539 unit uPSI_SynDBEdit; //Synedit
19420: 540 unit uPSI_SynEditWildcardSearch; //Synedit
19421: 541 unit uPSI_BoldComUtils; //BOLD
19422: 542 unit uPSI_BoldIsoDateTime; //BOLD
19423: 543 unit uPSI_BoldGUIDUtils; //inCOMUtils //BOLD
19424: 544 unit uPSI_BoldXMLRequests; //BOLD
19425: 545 unit uPSI_BoldStringList; //BOLD
19426: 546 unit uPSI_BoldFileHandler; //BOLD
19427: 547 unit uPSI_BoldContainers; //BOLD
19428: 548 unit uPSI_BoldQueryUserDlg; //BOLD
19429: 549 unit uPSI_BoldWinINet; //BOLD
19430: 550 unit uPSI_BoldQueue; //BOLD
19431: 551 unit uPSI_JvPcx; //JCL
19432: 552 unit uPSI_IdWhois; //Indy
19433: 553 unit uPSI_IdWhoIsServer; //Indy
19434: 554 unit uPSI_IdGopher; //Indy
19435: 555 unit uPSI_IdDateTimeStamp; //Indy
19436: 556 unit uPSI_IdDayTimeServer; //Indy
19437: 557 unit uPSI_IdDayTimeUDP; //Indy
19438: 558 unit uPSI_IdDayTimeUDPServer; //Indy
19439: 559 unit uPSI_IdDICTServer; //Indy
19440: 560 unit uPSI_IdDiscardServer; //Indy
19441: 561 unit uPSI_IdDiscardUDPServer; //Indy
19442: 562 unit uPSI_IdMappedFTP; //Indy
19443: 563 unit uPSI_IdMappedPortTCP; //Indy
19444: 564 unit uPSI_IdGopherServer; //Indy
19445: 565 unit uPSI_IdQotdServer; //Indy
19446: 566 unit uPSI_JvRgbToHtml; //JCL

```

```

19447: 567 unit uPSI_JvRemLog, //JCL
19448: 568 unit uPSI_JvSysComp; //JCL
19449: 569 unit uPSI_JvTMTL; //JCL
19450: 570 unit uPSI_JvWinampAPI; //JCL
19451: 571 unit uPSI_MSysUtils; //mX4
19452: 572 unit uPSI_ESBMaths; //ESB
19453: 573 unit uPSI_ESBMaths2; //ESB
19454: 574 unit uPSI_uLkJSON; //Lk
19455: 575 unit uPSI_ZURL; //Zeos
19456: 576 unit uPSI_ZSysUtils; //Zeos
19457: 577 unit unaUtils internals //UNA
19458: 578 unit uPSI_ZMatchPattern; //Zeos
19459: 579 unit uPSI_ZClasses; //Zeos
19460: 580 unit uPSI_ZCollections; //Zeos
19461: 581 unit uPSI_ZEncoding; //Zeos
19462: 582 unit uPSI_IdRawBase; //Indy
19463: 583 unit uPSI_IdNTLM; //Indy
19464: 584 unit uPSI_IdNNTP; //Indy
19465: 585 unit uPSI_usniffer; //PortScanForm
19466: 586 unit uPSI_IdCoderMIME; //Indy
19467: 587 unit uPSI_IdCoderUUE; //Indy
19468: 588 unit uPSI_IdCoderXXE; //Indy
19469: 589 unit uPSI_IdCoder3to4; //Indy
19470: 590 unit uPSI_IdCookie; //Indy
19471: 591 unit uPSI_IdCookieManager; //Indy
19472: 592 unit uPSI_WDosSocketUtils; //WDos
19473: 593 unit uPSI_WDosPlcUtils; //WDos
19474: 594 unit uPSI_WDosPorts; //WDos
19475: 595 unit uPSI_WDosResolvers; //WDos
19476: 596 unit uPSI_WDosTimers; //WDos
19477: 597 unit uPSI_WDosPlcs; //WDos
19478: 598 unit uPSI_WDosPneumatics; //WDos
19479: 599 unit uPSI_IdFingerServer; //Indy
19480: 600 unit uPSI_IdDNSResolver; //Indy
19481: 601 unit uPSI_IdHTTPWebBrokerBridge; //Indy
19482: 602 unit uPSI_IdIntercept; //Indy
19483: 603 unit uPSI_IdIPMCastBase; //Indy
19484: 604 unit uPSI_IdLogBase; //Indy
19485: 605 unit uPSI_IdIOHandlerStream; //Indy
19486: 606 unit uPSI_IdMappedPortUDP; //Indy
19487: 607 unit uPSI_IdQOTDUDPServer; //Indy
19488: 608 unit uPSI_IdQOTDUDP; //Indy
19489: 609 unit uPSI_IdSysLog; //Indy
19490: 610 unit uPSI_IdSysLogServer; //Indy
19491: 611 unit uPSI_IdSysLogMessage; //Indy
19492: 612 unit uPSI_IdTimeServer; //Indy
19493: 613 unit uPSI_IdTimeUDP; //Indy
19494: 614 unit uPSI_IdTimeUDPServer; //Indy
19495: 615 unit uPSI_IdUserAccounts; //Indy
19496: 616 unit uPSI_TextUtils; //mX4
19497: 617 unit uPSI_MandelbrotEngine; //mX4
19498: 618 unit uPSI_delphi_arduino_Unit1; //mX4
19499: 619 unit uPSI_DTDSchema2; //mX4
19500: 620 unit uPSI_fplotMain; //DMath
19501: 621 unit uPSI_FindFileIter; //mX4
19502: 622 unit uPSI_PppState; (JclStrHashMap) //PPP
19503: 623 unit uPSI_PppParser; //PPP
19504: 624 unit uPSI_PppLexer; //PPP
19505: 625 unit uPSI_PCharUtils; //PPP
19506: 626 unit uPSI_uJSON; //WU
19507: 627 unit uPSI_JclStrHashMap; //JCL
19508: 628 unit uPSI_JclHookExcept; //JCL
19509: 629 unit uPSI_EncdDecd; //VCL
19510: 630 unit uPSI_SockAppReg; //VCL
19511: 631 unit uPSI_PJFileHandle; //PJ
19512: 632 unit uPSI_PJEnvVars; //PJ
19513: 633 unit uPSI_PJPipe; //PJ
19514: 634 unit uPSI_PJPipeFilters; //PJ
19515: 635 unit uPSI_PJConsoleApp; //PJ
19516: 636 unit uPSI_UConsoleAppEx; //PJ
19517: 637 unit uPSI_DbxSocketChannelNative; //VCL
19518: 638 unit uPSI_DbxDataGenerator; //VCL
19519: 639 unit uPSI_DBXClient; //VCL
19520: 640 unit uPSI_IdLogEvent; //Indy
19521: 641 unit uPSI_Reversi; //mX4
19522: 642 unit uPSI_Geometry; //mX4
19523: 643 unit uPSI_IdSMTPServer; //Indy
19524: 644 unit uPSI_Textures; //mX4
19525: 645 unit uPSI_IBX; //VCL
19526: 646 unit uPSI_IWDBCommon; //VCL
19527: 647 unit uPSI_SortGrid; //mX4
19528: 648 unit uPSI_IB; //VCL
19529: 649 unit uPSI_IBScript; //VCL
19530: 650 unit uPSI_JvCSVBaseControls; //JCL
19531: 651 unit uPSI_Jvg3DColors; //JCL
19532: 652 unit uPSI_JvHLEditor; //beat
19533: 653 unit uPSI_JvShellHook; //JCL
19534: 654 unit uPSI_DBCommon2; //VCL
19535: 655 unit uPSI_JvSHFileOperation; //JCL

```

```

19536: 656 unit uPSI_uFileexport; //mX4
19537: 657 unit uPSI_JvDialogs; //JCL
19538: 658 unit uPSI_JvDBTreeView; //JCL
19539: 659 unit uPSI_JvDBUltimGrid; //JCL
19540: 660 unit uPSI_JvDBQueryParamsForm; //JCL
19541: 661 unit uPSI_JvExControls; //JCL
19542: 662 unit uPSI_JvBDEMemTable; //JCL
19543: 663 unit uPSI_JvCommStatus; //JCL
19544: 664 unit uPSI_JvMailSlots2; //JCL
19545: 665 unit uPSI_JvgWinMask; //JCL
19546: 666 unit uPSI_StEclipse; //SysTools
19547: 667 unit uPSI_StMime; //SysTools
19548: 668 unit uPSI_StList; //SysTools
19549: 669 unit uPSI_StMerge; //SysTools
19550: 670 unit uPSI_StStrS; //SysTools
19551: 671 unit uPSI_StTree; //SysTools
19552: 672 unit uPSI_StVArr; //SysTools
19553: 673 unit uPSI_StRegIni; //SysTools
19554: 674 unit uPSI_urkf; //DMath
19555: 675 unit uPSI_usvd; //DMath
19556: 676 unit uPSI_DepWalkUtils; //JCL
19557: 677 unit uPSI_OptionsFrm; //JCL
19558: 678 unit yuvconverts; //mX4
19559: 679 uPSI_JvPropAutoSave; //JCL
19560: 680 uPSI_AclAPI; //alcinoe
19561: 681 uPSI_AviCap; //alcinoe
19562: 682 uPSI_ALAVLBinaryTree; //alcinoe
19563: 683 uPSI_ALFcmMisc; //alcinoe
19564: 684 uPSI_ALStringList; //alcinoe
19565: 685 uPSI_ALQuickSortList; //alcinoe
19566: 686 uPSI_ALStaticText; //alcinoe
19567: 687 uPSI_ALJSONDoc; //alcinoe
19568: 688 uPSI_ALGSMComm; //alcinoe
19569: 689 uPSI_ALWindows; //alcinoe
19570: 690 uPSI_ALMultiPartFormDataParser; //alcinoe
19571: 691 uPSI_ALHttpCommon; //alcinoe
19572: 692 uPSI_ALWebSpider; //alcinoe
19573: 693 uPSI_ALHttpClient; //alcinoe
19574: 694 uPSI_ALFcmHTML; //alcinoe
19575: 695 uPSI_ALFTPClient; //alcinoe
19576: 696 uPSI_ALInternetMessageCommon; //alcinoe
19577: 697 uPSI_ALWininetHttpClient; //alcinoe
19578: 698 uPSI_ALWinInetFTPClient; //alcinoe
19579: 699 uPSI_ALWinHttpWrapper; //alcinoe
19580: 700 uPSI_ALWinHttpClient; //alcinoe
19581: 701 uPSI_ALFcmWinSock; //alcinoe
19582: 702 uPSI_ALFcmSQL; //alcinoe
19583: 703 uPSI_ALFcmCGI; //alcinoe
19584: 704 uPSI_ALFcmExecute; //alcinoe
19585: 705 uPSI_ALFcmFile; //alcinoe
19586: 706 uPSI_ALFcmMime; //alcinoe
19587: 707 uPSI_ALPhpRunner; //alcinoe
19588: 708 uPSI_ALGraphic; //alcinoe
19589: 709 uPSI_ALIniFiles; //alcinoe
19590: 710 uPSI_ALMemCachedClient; //alcinoe
19591: 711 unit uPSI_MyGrids; //mX4
19592: 712 uPSI_ALMultiPartMixedParser; //alcinoe
19593: 713 uPSI_ALSMTPClient; //alcinoe
19594: 714 uPSI_ALNNTPClient; //alcinoe
19595: 715 uPSI_ALHintBalloon; //alcinoe
19596: 716 unit uPSI_ALXmlDoc; //alcinoe
19597: 717 unit uPSI_IPCThrd; //VCL
19598: 718 unit uPSI_MonForm; //VCL
19599: 719 unit uPSI_TeCanvas; //Orpheus
19600: 720 unit uPSI_Ovcmisc; //Orpheus
19601: 721 unit uPSI_ovcfiler; //Orpheus
19602: 722 unit uPSI_ovcstate; //Orpheus
19603: 723 unit uPSI_ovccoco; //Orpheus
19604: 724 unit uPSI_ovcrvexp; //Orpheus
19605: 725 unit uPSI_OvcFormatSettings; //Orpheus
19606: 726 unit uPSI_OvcUtils; //Orpheus
19607: 727 unit uPSI_ovcstore; //Orpheus
19608: 728 unit uPSI_ovcstr; //Orpheus
19609: 729 unit uPSI_ovcmru; //Orpheus
19610: 730 unit uPSI_ovccmd; //Orpheus
19611: 731 unit uPSI_ovctimer; //Orpheus
19612: 732 unit uPSI_ovcintl; //Orpheus
19613: 733 uPSI_AfCircularBuffer; //AsyncFree
19614: 734 uPSI_AfUtils; //AsyncFree
19615: 735 uPSI_AfSafeSync; //AsyncFree
19616: 736 uPSI_AfComPortCore; //AsyncFree
19617: 737 uPSI_AfComPort; //AsyncFree
19618: 738 uPSI_AfPortControls; //AsyncFree
19619: 739 uPSI_AfDataDispatcher; //AsyncFree
19620: 740 uPSI_AfViewers; //AsyncFree
19621: 741 uPSI_AfDataTerminal; //AsyncFree
19622: 742 uPSI_SimplePortMain; //AsyncFree
19623: 743 unit uPSI_ovcclock; //Orpheus
19624: 744 unit uPSI_o32intlst; //Orpheus

```

```

19625: 745 unit uPSI_o32ledlabel; //Orpheus
19626: 746 unit uPSI_AlMySQLClient; //alcinoe
19627: 747 unit uPSI_ALFBXClient; //alcinoe
19628: 748 unit uPSI_ALFcnSQL; //alcinoe
19629: 749 unit uPSI_AsyncTimer; //mX4
19630: 750 unit uPSI_ApplicationFileIO; //mX4
19631: 751 unit uPSI_PsAPI; //VCLé
19632: 752 uPSI_ovcuser; //Orpheus
19633: 753 uPSI_ovcurl; //Orpheus
19634: 754 uPSI_ovcvlb; //Orpheus
19635: 755 uPSI_ovccolor; //Orpheus
19636: 756 uPSI_ALFBXLib; //alcinoe
19637: 757 uPSI_ovcmeter; //Orpheus
19638: 758 uPSI_ovcpeakm; //Orpheus
19639: 759 uPSI_O32BGSty; //Orpheus
19640: 760 uPSI_ovcBidi; //Orpheus
19641: 761 uPSI_ovctcary; //Orpheus
19642: 762 uPSI_DXPUtils; //mX4
19643: 763 uPSI_ALMultiPartBaseParser; //alcinoe
19644: 764 uPSI_ALMultiPartAlternativeParser; //alcinoe
19645: 765 uPSI_ALPOP3Client; //alcinoe
19646: 766 uPSI_SmallUtils; //mX4
19647: 767 uPSI_MakeApp; //mX4
19648: 768 uPSI_O32MouseMon; //Orpheus
19649: 769 uPSI_OvcCache; //Orpheus
19650: 770 uPSI_ovccalc; //Orpheus
19651: 771 uPSI_Joystick; //OpenGL
19652: 772 uPSI_ScreenSaver; //OpenGL
19653: 773 uPSI_XCollection; //OpenGL
19654: 774 uPSI_Polynomial; //OpenGL
19655: 775 uPSI_PersistentClasses; //9.86 //OpenGL
19656: 776 uPSI_VectorLists; //OpenGL
19657: 777 uPSI_XOpenGL; //OpenGL
19658: 778 uPSI_MeshUtils; //OpenGL
19659: 779 unit uPSI_JclSysUtils; //JCL
19660: 780 unit uPSI_JclBorlandTools; //JCL
19661: 781 unit JclFileUtils_max; //JCL
19662: 782 uPSI_AfDataControls; //AsyncFree
19663: 783 uPSI_GLSilhouette; //OpenGL
19664: 784 uPSI_JclSysUtils_class; //JCL
19665: 785 uPSI_JclFileUtils_class; //JCL
19666: 786 uPSI_FileUtil; //JCL
19667: 787 uPSI_changeFind; //mX4
19668: 788 uPSI_cmdIntf; //mX4
19669: 789 uPSI_fsService; //mX4
19670: 790 uPSI_Keyboard; //OpenGL
19671: 791 uPSI_VRMLParser; //OpenGL
19672: 792 uPSI_GLFileVRML; //OpenGL
19673: 793 uPSI_Octree; //OpenGL
19674: 794 uPSI_GLPolyhedron; //OpenGL
19675: 795 uPSI_GLCrossPlatform; //OpenGL
19676: 796 uPSI_GLParticles; //OpenGL
19677: 797 uPSI_GLNavigator; //OpenGL
19678: 798 uPSI_GLStarRecord; //OpenGL
19679: 799 uPSI_GLTextureCombiners; //OpenGL
19680: 800 uPSI_GLCanvas; //OpenGL
19681: 801 uPSI_GeometryBB; //OpenGL
19682: 802 uPSI_GeometryCoordinates; //OpenGL
19683: 803 uPSI_VectorGeometry; //OpenGL
19684: 804 uPSI_BumpMapping; //OpenGL
19685: 805 uPSI_TGA; //OpenGL
19686: 806 uPSI_GLVectorFileObjects; //OpenGL
19687: 807 uPSI_IMM; //VCL
19688: 808 uPSI_CategoryButtons; //VCL
19689: 809 uPSI_ButtonGroup; //VCL
19690: 810 uPSI_DbExcept; //VCL
19691: 811 uPSI_AxCtrls; //VCL
19692: 812 uPSI_GL_actorUnit1; //OpenGL
19693: 813 uPSI_StdVCL; //VCL
19694: 814 unit CurvesAndSurfaces; //OpenGL
19695: 815 uPSI_DataAwareMain; //AsyncFree
19696: 816 uPSI_TabNotBk; //VCL
19697: 817 uPSI_udwsfiler; //mX4
19698: 818 uPSI_synaip; //Synapse!
19699: 819 uPSI_synacode; //Synapse
19700: 820 uPSI_synachar; //Synapse
19701: 821 uPSI_synamisc; //Synapse
19702: 822 uPSI_synaser; //Synapse
19703: 823 uPSI_synaicnv; //Synapse
19704: 824 uPSI_tlntsend; //Synapse
19705: 825 uPSI_pingsend; //Synapse
19706: 826 uPSI_blksock; //Synapse
19707: 827 uPSI_asnlutil; //Synapse
19708: 828 uPSI_dnssend; //Synapse
19709: 829 uPSI_clamsend; //Synapse
19710: 830 uPSI_ldapsend; //Synapse
19711: 831 uPSI_mimemess; //Synapse
19712: 832 uPSI_slogsend; //Synapse
19713: 833 uPSI_mimepart; //Synapse

```



```

19714: 834 uPSI_mimeinln; //Synapse
19715: 835 uPSI_ftpsend, //Synapse
19716: 836 uPSI_ftptsend; //Synapse
19717: 837 uPSI_httpsend; //Synapse
19718: 838 uPSI_sntpsend; //Synapse
19719: 839 uPSI_smtpsend; //Synapse
19720: 840 uPSI_snmpsend; //Synapse
19721: 841 uPSI_imapsend; //Synapse
19722: 842 uPSI_pop3send; //Synapse
19723: 843 uPSI_nnptpsend; //Synapse
19724: 844 uPSI_ssl_cryptlib; //Synapse
19725: 845 uPSI_ssl_openssl; //Synapse
19726: 846 uPSI_synhttp_daemon; //Synapse
19727: 847 uPSI_NetWork; //mX4
19728: 848 uPSI_PingThread; //Synapse
19729: 849 uPSI_JvThreadTimer; //JCL
19730: 850 unit uPSI_wwSystem; //InfoPower
19731: 851 unit uPSI_IdComponent; //Indy
19732: 852 unit uPSI_IdIOHandlerThrottle; //Indy
19733: 853 unit uPSI_Themes; //VCL
19734: 854 unit uPSI_StdStyleActnCtrls; //VCL
19735: 855 unit uPSI_UDDIHelper; //VCL
19736: 856 unit uPSI_IdIMAP4Server; //Indy
19737: 857 uPSI_VariantSymbolTable, //VCL //3.9.9.92
19738: 858 uPSI_udf_glob, //mX4
19739: 859 uPSI_TabGrid, //VCL
19740: 860 uPSI_JsDBTreeView, //mX4
19741: 861 uPSI_JsSendMail, //mX4
19742: 862 uPSI_dbTvRecordList, //mX4
19743: 863 uPSI_TreeVwEx, //mX4
19744: 864 uPSI_ECDataLink, //mX4
19745: 865 uPSI_dbTree, //mX4
19746: 866 uPSI_dbTreeCBox, //mX4
19747: 867 unit uPSI_Debug; //TfrmDebug
19748: 868 uPSI_TimeFncs; //mX4
19749: 869 uPSI_FileIntf, //VCL
19750: 870 uPSI_SockTransport, //RTL
19751: 871 unit uPSI_WinInet; //RTL
19752: 872 unit uPSI_Wwstr; //mX4
19753: 873 uPSI_DBLookup, //VCL
19754: 874 uPSI_Hotspot, //mX4
19755: 875 uPSI_HList; //History List
19756: 876 unit uPSI_DrTable; //VCL
19757: 877 uPSI_TConnect, //VCL
19758: 978 uPSI_DataBkr, //VCL
19759: 979 uPSI_HTTPIntr; //VCL
19760: 980 unit uPSI_Mathbox; //mX4
19761: 881 uPSI_cyIndy, //cY
19762: 882 uPSI_cySysUtils, //cY
19763: 883 uPSI_cyWinUtils, //cY
19764: 884 uPSI_cyStrUtils, //cY
19765: 885 uPSI_cyObjUtils, //cY
19766: 886 uPSI_cyDateUtils, //cY
19767: 887 uPSI_cyBDE, //cY
19768: 888 uPSI_cyClasses, //cY
19769: 889 uPSI_cyGraphics, //3.9.9.94_2 //cY
19770: 890 unit uPSI_cyTypes; //cY
19771:
19772:
19773:
19774:
19775:
19776: //////////////////////////////////////
19777: //Form Template Library FTL
19778: //////////////////////////////////////
19779:
19780: 26 FTL For Form Building out of the Script, eg. 399_form_templates.txt
19781:
19782: 045 unit uPSI_VListView TFormListView;
19783: 263 unit uPSI_JvProfiler32; TProfReport
19784: 270 unit uPSI_imgList; TCustomImageList
19785: 278 unit uPSI_JvImageWindow; TJvImageWindow
19786: 317 unit uPSI_JvParserForm; TJvHTMLParserForm
19787: 497 unit uPSI_DebugBox; TDebugBox
19788: 513 unit uPSI_ImageWin; TImageForm, TImageForm2
19789: 514 unit uPSI_CustomDrawTreeView; TCustomDrawForm
19790: 515 unit uPSI_GraphWin; TGraphWinForm
19791: 516 unit uPSI_actionMain; TActionForm
19792: 518 unit uPSI_CtlPanel; TAppletApplication
19793: 529 unit uPSI_MDIEdit; TEditForm
19794: 535 unit uPSI_CoolMain; {browser} TWebMainForm
19795: 538 unit uPSI_frmExportMain; TSynexportForm
19796: 585 unit uPSI_usniffer; {PortScanForm} TSniffForm
19797: 600 unit uPSI_ThreadForm; TThreadSortForm;
19798: 618 unit uPSI_delphi_arduino_Unit1; TLEDForm
19799: 620 unit uPSI_fplotMain; TfplotForm1
19800: 660 unit uPSI_JvDBQueryParamsForm; TJvQueryParamsDialog
19801: 677 unit uPSI_OptionsFrm; TfrmOptions;
19802: 718 unit uPSI_MonForm; TMonitorForm

```

```

19803: 742 unit uPSI_SimplePortMain;
19804: 770 unit uPSI_ovccalc;
19805: 810 unit uPSI_DbExcept;
19806: 812 unit uPSI_GL_actorUnit1;
19807: 846 unit uPSI_synhttp_daemon;
19808: 867 unit uPSI_Debug;
19809:
19810:
19811: ex.:with TEditForm.create(self) do begin
19812:     caption:= 'Template Form Tester';
19813:     FormStyle:= fsStayOnTop;
19814:     with editor do begin
19815:         Lines.LoadFromFile(Exepath+'\docs\Readme_rus_mX2.rtf
19816:         SelStart:= 0;
19817:         Modified:= False;
19818:     end;
19819: end;
19820: with TWebMainForm.create(self) do begin
19821:     URLs.Text:= 'http://www.kleiner.ch';
19822:     URLsClick(self); Show;
19823: end;
19824: with TSynexportForm.create(self) do begin
19825:     Caption:= 'Synexport HTML RTF tester';
19826:     Show;
19827: end;
19828: with TThreadSortForm.create(self) do begin
19829:     showmodal; free;
19830: end;
19831:
19832: with TCustomDrawForm.create(self) do begin
19833:     width:=820; height:=820;
19834:     image1.height:= 600; //add properties
19835:     image1.picture.bitmap:= image2.picture.bitmap;
19836:     //SelectionBackground1Click(self) CustomDraw1Click(self);
19837:     Background1.click;
19838:     bitmap1.click;
19839:     Tile1.click;
19840:     Showmodal;
19841:     Free;
19842: end;
19843:
19844: with TfplotForm1.Create(self) do begin
19845:     BtnPlotClick(self);
19846:     Showmodal; Free;
19847: end;
19848:
19849: with TOvcCalculator.create(self) do begin
19850:     parent:= aForm;
19851:     //free;
19852:     setbounds(550,510,200,150);
19853:     displaystr:= 'maXcalc';
19854: end;
19855:
19856:
19857: //////////////////////////////////////
19858: All maXbox Tutorials Table of Content 2014
19859: //////////////////////////////////////
19860: Tutorial 00 Function-Coding (Blix the Programmer)
19861: Tutorial 01 Procedural-Coding
19862: Tutorial 02 OO-Programming
19863: Tutorial 03 Modular Coding
19864: Tutorial 04 UML Use Case Coding
19865: Tutorial 05 Internet Coding
19866: Tutorial 06 Network Coding
19867: Tutorial 07 Game Graphics Coding
19868: Tutorial 08 Operating System Coding
19869: Tutorial 09 Database Coding
19870: Tutorial 10 Statistic Coding
19871: Tutorial 11 Forms Coding
19872: Tutorial 12 SQL DB Coding
19873: Tutorial 13 Crypto Coding
19874: Tutorial 14 Parallel Coding
19875: Tutorial 15 Serial RS232 Coding
19876: Tutorial 16 Event Driven Coding
19877: Tutorial 17 Web Server Coding
19878: Tutorial 18 Arduino System Coding
19879: Tutorial 18_3 RGB LED System Coding
19880: Tutorial 19 WinCOM /Arduino Coding
19881: Tutorial 20 Regular Expressions RegEx
19882: Tutorial 21 Android Coding (coming 2013)
19883: Tutorial 22 Services Programming
19884: Tutorial 23 Real Time Systems
19885: Tutorial 24 Clean Code
19886: Tutorial 25 maXbox Configuration I+II
19887: Tutorial 26 Socket Programming with TCP
19888: Tutorial 27 XML & TreeView
19889: Tutorial 28 DLL Coding (available)
19890: Tutorial 29 UML Scripting (coming 2014)
19891: Tutorial 30 Web of Things (coming 2014)

```

```
19892: Tutorial 31 Closures (coming 2014)
19893: Tutorial 32 SQL Firebird (coming 2014)
19894:
19895:
19896: Doc ref Docu for all Type Class and Const in maXbox_types.pdf
19897: using Docu for this file is maxbox_functions_all.pdf
19898: PEP - Pascal Education Program Lib Lab
19899:
19900: http://stackoverflow.com/tags/pascalscript/hot
19901: http://www.jrsoftware.org/ishelp/index.php?topic=scriptfunctions
19902: #locs:15162
19903: http://sourceforge.net/apps/mediawiki/maXbox
19904: http://www.blaisepascal.eu/
19905: https://github.com/maxkleiner/maXbox3.git
19906: http://www.heise.de/download/maxbox-1176464.html
19907: http://www.softpedia.com/get/Programming/Other-Programming-Files/maXbox.shtml
19908: https://www.facebook.com/pages/Programming-maXbox/166844836691703
19909:
19910: ---- bigbitbox code_cleared_checked----
```