

```

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: 22463488 V3.9.9.98 August 2014 To EKON/BASTA/JAX/IBZ/SWS
10: *****Now the Funclist*****
11: Funclist Function : 12983 //10766 //10165 (7648)
12: *****Now the Proclist*****
13: Proclist Procedure Size is: 7997 //6792 //6401 4752
14: *****Now Constructors*****
15: Constructlist Constructor Size is: 1292 //995 //
16: def head:max: maxBox7: 23.07.2014 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: 22272! Constructor, Function and Procedure
21: AExtraxt of EXE Functions of maxbox3.exe, locs of file = 22020
22: ASize of EXE: 22463488 (22041600) (16586240) (13511680)
23: SHA1 Hash of maxbox 3.9.9.98: 2B7D1EBFB07DB9B0660F857320C44D70CAA1C717
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,OverlayImagIdx,
    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 AddParamSQLForDetail(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: TTextLineStyle', '(tlbsLF, tlbsCRLF)
109: Function AdjustLineBreaks(const S: string; Style: TTextLineStyle): 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 AnsiStrLIComp( 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 ASNDeclen( var Start : Integer; const Buffer : string) : Integer;
200: Function ASNDecOIDItem( 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( : 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 AUusername, 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 Bcd.ToDouble( 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 Binary.ToDouble( ABinary : string; DefValue : Double) : Double;
239: Function BinomialCoeff( N, R : Cardinal) : Float;
240: function BinomialCoefficient(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 BoolToStrl(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 : TStrings) : TStrings
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 : TStrings) : 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 CharIsEOF( 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 CheckBox: string;
344: Function CheckOpen( Status : DBIResult) : Boolean
345: Function CheckPassword( const APassword : String) : Boolean
346: Function CheckResponse(const AResponse:SmallInt;const AAllowedResponses:array of SmallInt): SmallInt
347: Function CheckCrc32( var X : array of Byte; N : Integer; Crc : Cardinal) : Integer;
348: function CheckSynchronize(Timeout: Integer): Boolean
349: Function CheckWin32Version( AMajor : Integer; AMinor : Integer) : Boolean
350: function ChrA(const a: byte): char;
351: Function ClassIDToProgID(const ClassID: TGUID): string;
352: Function ClassNameIs(const Name: string): Boolean
353: Function ClearBit( const Value : Byte; const Bit : TBitRange) : Byte;
354: Function ClearBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
355: Function ClearBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
356: Function ClearBit3( const Value : Word; const Bit : TBitRange) : Word;

```

```

357: Function ClearBit4( const Value : Integer; const Bit : TBitRange) : Integer;
358: Function ClearBit5( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
359: Function ClearBit6( const Value : Int64; const Bit : TBitRange) : Int64;
360: function CLIENTTOSCREEN(POINT:TPOINT):TPOINT
361: Function Clipboard : TClipboard
362: Function ClipCodes( const X, Y, MinX, MinY, MaxX, MaxY : Float) : TClipCodes;
363: Function ClipCodes1( const X, Y : Float; const ClipRect : TRect) : TClipCodes;
364: Function ClipLine( var X1, Y1, X2, Y2 : Integer; const ClipRect : TRect) : Boolean;
365: Function ClipLineToRect( var P1, P2 : TFloatPoint; const Rect : TFloatRect) : Boolean
366: Function Clone( out stm : IStream) : HResult
367: Function CloneConnection : TSQLConnection
368: Function CloneMemoryStream( Original : TMemoryStream) : TMemoryStream
369: function CLOSEQUERY:BOOLEAN
370: Function CloseVolume( var Volume : THandle) : Boolean
371: Function CloseHandle(Handle: Integer): Integer; stdcall;
372: Function CPLApplet( hwndCPL : THandle; uMsg : DWORD; lParam1, lParam2 : Longint) : Longint
373: Function CmdLine: PChar;
374: function CmdShow: Integer;
375: Function Color32( const R, G, B : Byte; const A : Byte) : TColor32;
376: Function Color32( WinColor : TColor) : TColor32;
377: Function Color32I( const Index : Byte; const Palette : TPalette32) : TColor32;
378: Function ColorAdjustLuma( clrRGB : TColor; n : Integer; fScale : Boolean) : TColor
379: Function ColorHLSToRGB( Hue, Luminance, Saturation : Word) : TColorRef
380: Function ColorToHTML( const Color : TColor) : String
381: function ColorToIdent(Color: Longint; var Ident: string): Boolean
382: Function ColorToRGB(color: TColor): Longint
383: function ColorToString(Color: TColor): string
384: Function ColorToWebColorName( Color : TColor) : string
385: Function ColorToWebColorStr( Color : TColor) : string
386: Function ColumnAtDepth( Col : TColumn; ADepth : Integer) : TColumn
387: Function Combination(npr, ncr: integer): extended;
388: Function CombinationInt(npr, ncr: integer): Int64;
389: Function CombineInfo( Bitmap : TCustomBitmap32) : TCombineInfo
390: Function CommaAdd( const AStr1, AStr2 : String) : string
391: Function CommercialRound( const X : Extended) : Int64
392: Function Commit( grfCommitFlags : Longint) : HResult
393: Function Compare( const NameExt : string) : Boolean
394: function CompareDate(const A, B: TDateTime): TValueRelationship;
395: Function CompareDateTime( const ADateTime1, ADatetime2 : TDateTime) : Integer
396: Function CompareFiles(const FN1, FN2 : string; Breathe:TNotifyEvent;BreathingSender:TObject) : boolean
397: Function CompareMemoryStreams( S1, S2 : TMemoryStream) : boolean
398: Function CompareStr( S1, S2 : string) : Integer
399: function CompareStr(const S1: string; const S2: string): Integer
400: function CompareString(const S1: string; const S2: string): Integer
401: Function CompareText( S1, S2 : string) : Integer
402: function CompareText(const S1: string; const S2: string): Integer
403: Function CompareTextLike(cWildStr,cStr:string;const cWildChar:char;lCaseSensitive:boolean): boolean
404: function CompareTime( const A, B: TDateTime): TValueRelationship;
405: Function CompatibleConversionType( const AType : TConvType; const AFamily : TConvFamily) : Boolean
406: Function CompatibleConversionTypes( const AFrom, ATo : TConvType) : Boolean
407: Function ComponentTypeToString( const ComponentType : DWORD) : string
408: Function CompToCurrency( Value : Comp) : Currency
409: Function Comp.ToDouble( Value : Comp) : Double
410: function ComputeFileCRC32(const FileName : String) : Integer;
411: function ComputeSHA256(astr: string; amode: char): string //mode F:File, S:String
412: function ComputeSHA512(astr: string; amode: char): string //mode F:File, S:String
413: Function Concat(s: string): string
414: Function ConnectAndGetAll : string
415: Function Connected : Boolean
416: function constrain(x, a, b: integer): integer;
417: Function ConstraintCallBack( Req : DsInfoReq; var ADataSources : DataSources) : DBIResult
418: Function ConstraintsDisabled : Boolean
419: function CONTAINSCONTROL(CONTROL:TCONTROL):BOOLEAN
420: Function ContainsState( oState : ThiRegularExpressionState) : boolean
421: Function ContainsStr( const AText, ASubText : string) : Boolean
422: Function ContainsText( const AText, ASubText : string) : Boolean
423: Function ContainsTransition( oTransition : ThiRegularExpressionTransition) : boolean
424: Function Content : string
425: Function ContentFromStream( Stream : TStream) : string
426: Function ContentFromString( const S : string) : string
427: Function CONTROLSDISABLED : BOOLEAN
428: Function Convert( const AValue : Double; const AFrom, ATo : TConvType) : Double;
429: Function Convert1( const AValue : Double; const AFrom1, AFrom2, ATo1, ATo2 : TConvType) : Double;
430: Function ConvertFrom( const AFrom : TConvType; const AValue : Double) : Double
431: Function ConvertReadStream( Stream : TStream; Buffer : PChar; BufSize : Integer) : Integer
432: Function ConvertTo( const AValue : Double; const ATo : TConvType) : Double
433: Function ConvertWriteStream( Stream : TStream; Buffer : PChar; Bufsize : Integer) : Integer
434: Function ConvFamilyToDescription( const AFamily : TConvFamily) : string
435: Function ConvTypeToDescription( const AType : TConvType) : string
436: Function ConvtypeToFamily( const AFrom, ATo : TConvType) : TConvFamily;
437: Function ConvTypeToFamily( const AType : TConvType) : TConvFamily;
438: Function ConvAdd(const AVal:Double;const AType1:TConvType;const AVal2:Double;const AType2,
AResultType:TConvType): Double
439: Function ConvCompareValue(const AValue1:Double;const AType1:TConvType;const AValue2:Double;const
AType2:TConvType): TValueRelationship
440: Function ConvDec( const AValue : Double; const AType, AAmountType : TConvType) : Double;
441: Function ConvDecl( const AValue:Double;const AType: TConvType;const AAmount:Double;const
AAmountType:TConvType):Double;
442: Function ConvDiff(const AVal1:Double;const AType1:TConvType;const AVal2:Double;const AType2,
AResuType:TConvType):Double

```

```

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

```

```

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

```

```

613: Function DelChar( const pStr : string; const pChar : Char ) : string
614: Function DelEnvironmentVar( const Name : string ) : Boolean
615: Function Delete( const MsgNum : Integer ) : Boolean
616: Function DeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean ) : Boolean
617: Function DeleteFile( const FileName : string ) : boolean
618: Function DeleteFileEx( FileName : string; Flags : FILEOP_FLAGS ) : Boolean
619: Function DelimiterPosn( const sString : string; const sDelimiters: string): integer;
620: Function DelimiterPosnl(const sString:string;const sDelimiters:string;out cDelimiter: char) : integer;
621: Function DelSpace( const pStr : string ) : string
622: Function DelString( const pStr, pDelStr : string ) : string
623: Function DelTree( const Path : string ) : Boolean
624: Function Depth : Integer
625: Function Description : string
626: Function DescriptionsAvailable : Boolean
627: Function DescriptionToConvFamily( const ADescription : string; out AFamily : TConvFamily ) : Boolean
628: Function DescriptionToConvType( const ADescription : string; out AType : TConvType ) : Boolean;
629: Function DescriptionToConvTypel(const AFamil:TConvFamily;const ADescr:string;out AType:TConvType):Boolean;
630: Function DetectUTF8Encoding( const s : UTF8String ) : TEcodeType
631: Function DialogsTopixelsX( const Dialogs : Word ) : Word
632: Function DialogsTopixelsY( const Dialogs : Word ) : Word
633: Function Digits( const X : Cardinal ) : Integer
634: Function DirectoryExists( const Name : string ) : Boolean
635: Function DirectoryExists( Directory : string ) : Boolean
636: Function DiskFree( Drive : Byte ) : Int64
637: function DiskFree(Drive: Byte): Int64)
638: Function DiskInDrive( Drive : Char ) : Boolean
639: Function DiskSize( Drive : Byte ) : Int64
640: function DiskSize(Drive: Byte): Int64)
641: Function DISPATCHCOMMAND( ACOMMAND : WORD ) : BOOLEAN
642: Function DispatchEnabled : Boolean
643: Function DispatchMask : TMask
644: Function DispatchMethodType : TMethodType
645: Function DISPATCHPOPUP( AHANDLE : HMENU ) : BOOLEAN
646: Function DispatchRequest( Sender : TObject; Request : TWebRequest; Response : TWebResponse ) : Boolean
647: Function DisplayCase( const S : String ) : String
648: Function DisplayRect( Code : TDisplayCode ) : TRect
649: Function DisplayRect( TextOnly : Boolean ) : TRect
650: Function DisplayStream( Stream : TStream ) : string
651: TBufferCoord', 'record Char : integer; Line : integer; end
652: TDisplayCoord', 'record Column : integer; Row : integer; end
653: Function DisplayCoord( AColumn, ARow : Integer ) : TDisplayCoord
654: Function BufferCoord( AChar, ALine : Integer ) : TBufferCoord
655: Function DomainName( const AHost : String ) : String
656: Function DosPathToUnixPath( const Path : string ) : string
657: Function DottedLineTo( const Canvas : TCanvas; const X, Y : Integer ) : Boolean;
658: Function DoubleDecliningBalance( const Cost, Salvage : Extended; Life, Period : Integer ) : Extended
659: Function DoubleToBcd( const AValue : Double ) : TBcd;
660: Function DoubleToHex( const D : Double ) : string
661: Function DoUpdates : Boolean
662: function Dragging: Boolean;
663: Function DrawCaption( pl : HWND; p2 : HDC; const p3 : TRect; p4 : UInt ) : BOOL
664: Function DrawAnimatedRects( hwnd : HWND; idAni : Integer; const lprcFrom, lprcTo : TRect ) : BOOL
665: Function DrawEdge( hdc : HDC; var qrc : TRect; edge : UInt; grfFlags : UInt ) : BOOL
666: Function DrawFrameControl( DC : HDC; const Rect : TRect; uType, uState : UInt ) : BOOL
667: {Works like InputQuery but displays 2 edits. If PasswordChar <> #0, the second edit's PasswordChar is set}
668: Function DualInputQuery(const ACapt,Prpt1,Prpt2:string;var AVall,AVal2:string;PasswrDChar:Char= #0):Bool;
669: Function DupeString( const AText : string; ACount : Integer ) : string
670: Function Edit : Boolean
671: Function EditCaption : Boolean
672: Function EditText : Boolean
673: Function EditFolderList( Folders : TStrings ) : Boolean
674: Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext ) : Boolean
675: Function Elapsed( const Update : Boolean ) : Cardinal
676: Function EnableProcessPrivilege( const Enable : Boolean; const Privilege : string ) : Boolean
677: Function EnableThreadPrivilege( const Enable : Boolean; const Privilege : string ) : Boolean
678: Function EncodeDate( Year, Month, Day : Word ) : TDateTime
679: function EncodeDate(Year, Month, Day: Word): TDateTime;
680: Function EncodeDateDay( const AYear, ADayOfYear : Word ) : TDateTime
681: Function EncodeDateMonthWeek( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word ) : TDateTime
682: Function EncodeDateTime( const AYear, AMonth, ADay, AHour, AMinute, ASecond, AMilliSecond: Word): TDateTime
683: Function EncodeDateWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word ) : TDateTime
684: Function EncodeDayOfWeekInMonth( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word ) : TDateTime
685: Function EncodeString( s : string ) : string
686: Function DecodeString( s : string ) : string
687: Function EncodeTime( Hour, Min, Sec, MSec : Word ) : TDateTime
688: function EncodeTime(Hour, Min, Sec, MSec: Word): TDateTime;
689: Function EndIP : String
690: Function EndOfDay( const AYear, AMonth, ADay : Word ) : TDateTime;
691: Function EndOfDay1( const AYear, ADayOfYear : Word ) : TDateTime;
692: Function EndOfAMonth( const AYear, AMonth : Word ) : TDateTime
693: Function EndOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word ) : TDateTime
694: Function EndOfAYear( const AYear : Word ) : TDateTime
695: Function EndOfTheDay( const AValue : TDateTime ) : TDateTime
696: Function EndOfTheMonth( const AValue : TDateTime ) : TDateTime
697: Function EndOfTheWeek( const AValue : TDateTime ) : TDateTime
698: Function EndOfTheYear( const AValue : TDateTime ) : TDateTime
699: Function EndPeriod( const Period : Cardinal ) : Boolean
700: Function EndsStr( const ASubText, AText : string ) : Boolean
701: Function EndsText( const ASubText, AText : string ) : Boolean

```

```

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

```

```

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

```

```

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

```

```

963: Function FormatValue( AValue : Cardinal ) : string
964: Function FormatVersionString( const HiV, LoV : Word ) : string;
965: Function FormatVersionString1( const Major, Minor, Build, Revision : Word ) : string;
966: function Frac(X: Extended): Extended;
967: Function FreeResource( ResData : HGLOBAL ) : LongBool
968: Function FromCommon( const AValue : Double ) : Double
969: function FromCommon(const AValue: Double): Double;
970: Function FTPGMTDateToMLS( const ATimestamp : TDateTime; const AIIncludeMSecs : Boolean ) : String
971: Function FTPLocalDateToMLS( const ATimestamp : TDateTime; const AIIncludeMSecs : Boolean ) : String
972: Function FTPMLSToGMTDateTime( const ATimestamp : String ) : TDateTime
973: Function FTPMLSToLocalDateTime( const ATimestamp : String ) : TDateTime
974: Function FuncIn(AValue: Variant; ASet: Variant): Boolean;
975: //Function FuncIn Size is: 6444 of mX3.9.8.9
976: Function FutureValue(const Rate:Extended;NPeriods:Integer;const Payment,PresentValue:Extended;PaymentTime:TPaymentTime):Extended
977: Function Gauss( const x, Spread : Double ) : Double
978: function Gauss(const x,Spread: Double): Double;
979: Function GCD(x, y : LongInt) : LongInt;
980: Function GCDJ( X, Y : Cardinal ) : Cardinal
981: Function GDAL: LongWord
982: Function GdiFlush : BOOL
983: Function GdiSetBatchLimit( Limit : DWORD ) : DWORD
984: Function GdiGetBatchLimit : DWORD
985: Function GenerateHeader : TIdHeaderList
986: Function GeometricMean( const X : TDynFloatArray ) : Float
987: Function Get( AURL : string ) : string;
988: Function Get2( AURL : string ) : string;
989: Function Get8087CW : Word
990: function GetActiveOleObject(const ClassName: String): IDispatch;
991: Function GetAliasDriverName( const AliasName : string ) : string
992: Function GetAPMBatteryFlag : TAPMBatteryFlag
993: Function GetAPMBatteryFullLifeTime : DWORD
994: Function GetAPMBatteryLifePercent : Integer
995: Function GetAPMBatteryLifeTime : DWORD
996: Function GetAPMLineStatus : TAPMLineStatus
997: Function GetAppdataFolder : string
998: Function GetAppDispatcher : TComponent
999: function GetArrayLength: integer;
1000: Function GetASCII: string;
1001: Function GetASCIILine: string;
1002: Function GetAsHandle( Format : Word ) : THandle
1003: function GetAssociatedProgram(const Extension: string; var Filename, Description: string): boolean;
1004: Function GetBackupfileName( const FileName : string ) : string
1005: Function GetBBitmap( Value : TBitmap ) : TBitmap
1006: Function GetBIOSCopyright : string
1007: Function GetBIOSDate : TDateTime
1008: Function GetBIOSExtendedInfo : string
1009: Function GetBIOSName : string
1010: Function getBitmap(apath: string): TBitmap;
1011: Function GetBitmap( Index : Integer; Image : TBitmap ) : Boolean //object
1012: Function getBitMapObject(const bitmappath: string): TBitmap;
1013: Function GetButtonState( Button : TPageScrollerButton ) : TPageScrollerButtonState
1014: Function GetCapsLockKeyState : Boolean
1015: function GetCaptureControl: TControl;
1016: Function GetCDAudioTrackList( var TrackList : TJclCdTrackInfoArray; Drive : Char ) : TJclCdTrackInfo;
1017: Function GetCDAudioTrackList1( TrackList : TStrings; IncludeTrackType : Boolean; Drive : Char ) : string;
1018: Function GetCdInfo( InfoType : TJclCdMediaInfo; Drive : Char ) : string
1019: Function GetChangedText( const Text : string; SelStart, SelLength : Integer; Key : Char ) : string
1020: Function GetClientThread( ClientSocket : TServerClientWinSocket ) : TServerClientThread
1021: Function GetClockValue : Int64
1022: function getCmdLine: PChar;
1023: function getCmdShow: Integer;
1024: function GetCPUSpeed: Double;
1025: Function GetColField( DataCol : Integer ) : TField
1026: Function GetColorBlue( const Color : TColor ) : Byte
1027: Function GetColorFlag( const Color : TColor ) : Byte
1028: Function GetColorGreen( const Color : TColor ) : Byte
1029: Function GetColorRed( const Color : TColor ) : Byte
1030: Function GetComCtlVersion : Integer
1031: Function GetComPorts: TStringlist;
1032: Function GetCommonAppdataFolder : string
1033: Function GetCommonDesktopdirectoryFolder : string
1034: Function GetCommonFavoritesFolder : string
1035: Function GetCommonFilesFolder : string
1036: Function GetCommonProgramsFolder : string
1037: Function GetCommonStartmenuFolder : string
1038: Function GetCommonStartupFolder : string
1039: Function GetComponent( Owner, Parent : TComponent ) : TComponent
1040: Function GetConnectionRegistryFile( DesignMode : Boolean ) : string
1041: Function GetCookiesFolder : string
1042: Function GetCPUSpeed( var CpuSpeed : TFreqInfo ) : Boolean
1043: Function GetCurrent : TFavoriteLinkItem
1044: Function GetCurrent : TListItem
1045: Function GetCurrent : TTaskDialogBaseButtonItem
1046: Function GetCurrent : TToolButton
1047: Function GetCurrent : TTreenode
1048: Function GetCurrent : WideString
1049: Function GetCurrentDir : string
1050: function GetCurrentDir: string)

```

```

1051: Function GetCurrentFolder : string
1052: Function GETCURRENTRECORD( BUFFER : PCHAR ) : BOOLEAN
1053: Function GetCurrentProcessId : TIdPID
1054: Function GetCurrentThreadHandle : THandle
1055: Function GetCurrentThreadID: LongWord; stdcall;
1056: Function GetCustomHeader( const Name : string ) : String
1057: Function GetDataItem( Value : Pointer ) : Longint
1058: Function GetDataLinkFiles( FileNames : TWideStrings; Directory : string ) : Integer;
1059: Function GetDataLinkFiles1( FileNames : TStrings; Directory : string ) : Integer;
1060: Function GETDATASIZE : INTEGER
1061: Function GetDC(hdwnd: HWND): HDC;
1062: Function GetDefaultFileExt( const MIMEType : string ) : string
1063: Function GetDefaults : Boolean
1064: Function GetDefaultSchemaName : WideString
1065: Function GetDefaultStreamLoader : IStreamLoader
1066: Function GetDesktopDirectoryFolder : string
1067: Function GetDesktopFolder : string
1068: Function GetDFAState( oStates : TList ) : TniRegularExpressionState
1069: Function GetDirectorySize( const Path : string ) : Int64
1070: Function GetDisplayWidth : Integer
1071: Function GetDLLVersion( const DLLName : string; var pdwMajor, pdwMinor : Integer ) : Boolean
1072: Function GetDomainName : string
1073: Function GetDriverRegistryFile( DesignMode : Boolean ) : string
1074: function GetDriveType(rootpath: pchar): cardinal;
1075: Function GetDriveTypeStr( const Drive : Char ) : string
1076: Function GetEnumerator : TFavoriteLinkItemsEnumerator
1077: Function GetEnumerator : TListItemsEnumerator
1078: Function GetEnumerator : TTaskDialogButtonsEnumerator
1079: Function GetEnumerator : TToolBarEnumerator
1080: Function GetEnumerator : TTreeNodesEnumerator
1081: Function GetEnumerator : TWideStringsEnumerator
1082: Function GetEnvVar( const VarName : string ) : string
1083: Function GetEnvironmentVar( const AVariableName : string ) : string
1084: Function GetEnvironmentVariable( const VarName : string ) : string
1085: Function GetEnvironmentVar( const Name : string; var Value : string; Expand : Boolean ) : Boolean
1086: Function GetEnvironmentVars( const Vars : TStrings; Expand : Boolean ) : Boolean
1087: Function getEnvironmentString: string;
1088: Function GetExceptionHandler : TObject
1089: Function GetFavoritesFolder : string
1090: Function GetFieldByName( const Name : string ) : string
1091: Function GetFieldInfo( const Origin : Widestring; var FieldInfo : TFieldInfo ) : Boolean
1092: Function GetFieldValue( ACol : Integer ) : string
1093: Function GetFileAgeCoherence( const FileName : string ) : Boolean
1094: Function GetFileCreation( const FileName : string ) : TFileTime
1095: Function GetFileCreationTime( const Filename : string ) : TDateTime
1096: Function GetFileInfo( const FileName : string ) : TSearchRec
1097: Function GetFileLastAccess( const FileName : string ) : TFileTime
1098: Function GetFileLastWrite( const FileName : string ) : TFileTime
1099: Function GetFileList(FileList: TStringlist; apath: string): TStringlist;
1100: Function GetFileList1(apath: string): TStringlist;
1101: Function GetFileMIMEType( const AFileName : string ) : string
1102: Function GetFileSize( const FileName : string ) : Int64
1103: Function GetFileVersion( AFileName : string ) : Cardinal
1104: Function GetFileVersion( const AFilename : string ) : Cardinal
1105: Function GetFileSize2(Handle: Integer; x: Integer): Integer; stdcall;
1106: Function GetFileDate(aFile:string; aWithTime:Boolean):string;
1107: Function GetFilterData( Root : PExprNode ) : TExprData
1108: Function getChild : LongInt
1109: Function getChild : TTreeNode
1110: Function GetFirstDelimitedToken( const cDelim : char; const cStr : string ) : string
1111: Function GetFirstNode : TTreeNode
1112: Function GetFontsFolder : string
1113: Function GetFormulaValue( const Formula : string ) : Extended
1114: Function GetFreePageFileMemory : Integer
1115: Function GetFreePhysicalMemory : Integer
1116: Function GetFreeSystemResources( const ResourceType : TFreeSysResKind ) : Integer;
1117: Function GetFreeSystemResources1 : TFreeSystemResources;
1118: Function GetFreeVirtualMemory : Integer
1119: Function GetFromClipboard : Boolean
1120: Function GetFullURI( const AOptionalFields : TIdURIOptionalFieldsSet ) : string
1121: Function GetGBitmap( Value : TBitmap ) : TBitmap
1122: Function GetGMTDateByName( const AFileName : TIdFileName ) : TDateTime
1123: Function GetGroupState( Level : Integer ) : TGroupPosInds
1124: Function GetHandle : HWND
1125: Function GETHELPCONTEXT( VALUE : INTEGER; BYCOMMAND : BOOLEAN ) : THELPCONTEXT
1126: function GetHexArray(ahexdig: THexArray): THexArray;
1127: Function GetHighLightColor( const Color : TColor; Luminance : Integer ) : TColor
1128: function GetHINSTANCE: longword;
1129: Function GetHistoryFolder : string
1130: Function GetHitTestInfoAt( X, Y : Integer ) : THitTests
1131: function getHMODULE: longword;
1132: Function GetHostName(const AComputerName: String): String;
1133: Function GetHostName : string
1134: Function getHostIP: string;
1135: Function GetHotSpot : TPoint
1136: Function GetHueBitmap( Value : TBitmap ) : TBitmap
1137: Function GetImageBitmap : HBITMAP
1138: Function GETIMAGELIST : TCUSTOMIMAGELIST
1139: Function GetIncome( const aNetto : Currency ) : Currency

```

```

1140: Function GetIncome( const aNetto : Extended ) : Extended
1141: Function GetIncome( const aNetto : Extended): Extended
1142: Function GetIncome(const aNetto : Extended) : Extended
1143: function GetIncome(const aNetto: Currency): Currency
1144: Function GetIncome2( const aNetto : Currency) : Currency
1145: Function GetIncome2( const aNetto : Currency): Currency
1146: Function getIndex_Attrs( tag : string; var idx : Integer; var Attrs : string ) : string
1147: Function GETINDEXFORFIELDS( const FIELDS : String; CASEINSENSITIVE : BOOLEAN ) : TINDEXDEF
1148: Function GetIndexForOrderBy( const SQL : WideString; DataSet : TDataSet ) : TIndexDef
1149: Function GetInstRes(Instance:THandle;ResType:TResType;const
Name:string;Width:Integer;LoadFlags:TLoadResources;MaskColor:TColor):Boolean;
1150: Function
GetInstRes1(Instance:THandle;ResType:TResType;ResID:DWORD;Width:Int;LoadFlags:TLoadResources;MaskColor:TColor):Bool;
1151: Function GetIntelCacheDescription( const D : Byte ) : string
1152: Function GetInteractiveUserName : string
1153: Function GetInternetCacheFolder : string
1154: Function GetInternetFormattedFileTimeStamp( const AFilename : String ) : String
1155: Function GetIPAddress( const HostName : string ) : string
1156: Function GetIP( const HostName : string ) : string
1157: Function GetIPHostByName(const AComputerName: String): String;
1158: Function GetIsAdmin: Boolean;
1159: Function GetItem( X, Y : Integer ) : LongInt
1160: Function GetItemAt( X, Y : Integer ) : TListItem
1161: Function GetItemHeight(Font: TFont): Integer;
1162: Function GetItemPath( Index : Integer ) : string
1163: Function GetKeyFieldNames( List : TStrings ) : Integer;
1164: Function GetKeyFieldNames1( List : TWideStrings ) : Integer;
1165: Function GetKeyState( const VirtualKey : Cardinal ) : Boolean
1166: Function GetLastChild : LongInt
1167: Function GetLastChild : TTreeNode
1168: Function GetLastDelimitedToken( const cDelim : char; const cStr : string ) : string
1169: function GetLastError: Integer
1170: Function GetLAT_CONV_FACTOR: double; //for WGS84 power(1 - 1 / 298.257223563, 2);
1171: Function GetLoader( Ext : string ) : TBitmapLoader
1172: Function GetLoadFilter : string
1173: Function GetLocalComputerName : string
1174: Function GetLocaleChar( Locale, LocaleType : Integer; Default : Char ) : Char
1175: Function GetLocaleStr( Locale, LocaleType : Integer; Default : string ) : string
1176: Function GetLocalUserName : string
1177: Function GetLoginUsername : WideString
1178: function getLongDayNames: string)
1179: Function GetLongHint(const hint: string): string
1180: function getLongMonthNames: string)
1181: Function GetMacAddresses( const Machine : string; const Addresses : TStrings ) : Integer
1182: Function GetMainAppWndFromPid( PID : DWORD ) : HWND
1183: Function GetMaskBitmap : HBITMAP
1184: Function GetMaxAppAddress : Integer
1185: Function GetMciErrorMessage( const MciErrNo : MCIERROR ) : string
1186: Function GetMemoryLoad : Byte
1187: Function GetMIMEDefaultFileExt( const MIMEType : string ) : TIdFileName
1188: Function GetMIMETypeFromFile( const AFile : string ) : string
1189: Function GetMIMETypeFromFileName( const AFile : TIdFileName ) : string
1190: Function GetMinAppAddress : Integer
1191: Function GetModule : TComponent
1192: Function GetModuleHandle( ModuleName : PChar ) : HMODULE
1193: Function GetModuleName( Module : HMODULE ) : string
1194: Function GetModulePath( const Module : HMODULE ) : string
1195: Function GetModuleFileName(Module: Integer; Filename: PChar;Size: Integer): Integer; stdcall;
1196: Function GetCommandLine: PChar; stdcall;
1197: Function GetMonochromeBitmap( Value : TBitmap ) : TBitmap
1198: Function GetMultiN(aval: integer): string;
1199: Function GetName : String
1200: Function GetNearestItem( Point : TPoint; Direction : TSearchDirection ) : TListItem
1201: Function GetNethoodFolder : string
1202: Function GetNext : TTreeNode
1203: Function GetNextChild( Value : LongInt ) : LongInt
1204: Function GetNextChild( Value : TTreeNode ) : TTreeNode
1205: Function GetNextDelimitedToken( const cDelim : char; var cStr : String ) : String
1206: Function GetNextItem( StartItem: TListItem;Direction:TSearchDirection;States:TItemStates ) : TListItem
1207: Function GetNextPacket : Integer
1208: Function getNextSibling : TTreeNode
1209: Function GetNextVisible : TTreeNode
1210: Function GetNode( ItemId : HTreeItem ) : TTreeNode
1211: Function GetNodeAt( X, Y : Integer ) : TTreeNode
1212: Function GetNodeDisplayWidth( Node : TOutlineNode ) : Integer
1213: function GetNumberOfProcessors: longint;
1214: Function GetNumLockKeyState : Boolean
1215: Function GetObjectProperty( Instance : TPersistent; const PropName : string ) : TObject
1216: Function GetOnlyTransitionOn( cChar : char ) : TniRegularExpressionState
1217: Function GetOptionalParam( const ParamName : string ) : OleVariant
1218: Function GetOSName: string;
1219: Function GetOSVersion: string;
1220: Function GetOSNumber: string;
1221: Function GetOsVersionInfo: TOSVersionInfo; //thx to wischnewski
1222: Function GetPackageModuleHandle( PackageName : PChar ) : HMODULE
1223: function GetPageSize: Cardinal;
1224: Function GetParameterFileName : string
1225: Function GetParams( var OwnerData : OleVariant ) : OleVariant
1226: Function GETPARENTCOMPONENT : TCOMPONENT

```

```

1227: Function GetParentForm(control: TControl): TForm
1228: Function GETPARENTMENU : TMENU
1229: Function GetPassword : Boolean
1230: Function GetPassword : string
1231: Function GetPersonalFolder : string
1232: Function GetPidFromProcessName( const ProcessName : string ) : DWORD
1233: Function GetPosition : TPoint
1234: Function GetPrev : TTreeNode
1235: Function GetPrevChild( Value : LongInt ) : LongInt
1236: Function GetPrevChild( Value : TTreeNode ) : TTreeNode
1237: Function getPrevSibling : TTreeNode
1238: Function GetPrevVisible : TTreeNode
1239: Function GetPrinthoodFolder : string
1240: Function GetPrivilegeDisplayName( const PrivilegeName : string ) : string
1241: Function getProcessList: TStrings;
1242: Function GetProcessId : TIdPID
1243: Function GetProcessNameFromPid( PID : DWORD ) : string
1244: Function GetProcessNameFromWnd( Wnd : HWND ) : string
1245: Function GetProcessMemoryInfo(Process: THandle; ppsmemCounters: TProcessMemoryCounters; cb: DWORD): BOOL
1246: Function getProcessAllMemory(ProcessID : DWORD): TProcessMemoryCounters;
1247: Function getProcessMemoryInfo2(ProcessID : DWORD): TProcessMemoryCounters;
1248: Function GetProgramFilesPolder : string
1249: Function GetProgramsFolder : string
1250: Function GetProxy : string
1251: Function GetQuoteChar : WideString
1252: Function GetQrCode4(Width,Height:Word; Correct_Level:string;const Data:string;aformat:string):TLinearBitmap;
1253: Function GetQrCodetoFile(Width,Height:Word;Correct_Level:string;const Data:string;aformat:string):TLinearBitmap;
1254: Function GetRate : Double
1255: Function getPerfTime: string;
1256: Function getRuntime: string;
1257: Function GetRBitmap( Value : TBitmap ) : TBitmap
1258: Function GetReadableName( const AName : string ) : string
1259: Function GetRecentDocs : TStringList
1260: Function GetRecentFolder : string
1261: Function GetRecords( Count : Integer; out RecsOut : Integer; Options : Integer ) : OleVariant;
1262: Function GetRecords1(Count:Integer; out RecsOut:Integer;Options:Integer;const CommandText:WideString;var Params, OwnerData : OleVariant) : OleVariant;
1263: Function GetRecordset( const CommandText : WideString; ConnectionString : WideString ) : _Recordset
1264: Function GetRegisteredCompany : string
1265: Function GetRegisteredOwner : string
1266: Function GetResource(ResType:TResType;const Name:string;Width:Int;LoadFlags:TLoadResources;MaskColor:TColor:Bool
1267: Function GetResourceName( ObjStream : TStream; var AName : string ) : Boolean
1268: Function GetResponse( const AAllowedResponses : array of SmallInt ) : SmallInt;
1269: Function GetResponse1( const AAllowedResponse : SmallInt ) : SmallInt;
1270: Function GetRValue( rgb : DWORD ) : Byte
1271: Function GetGValue( rgb : DWORD ) : Byte
1272: Function GetBValue( rgb : DWORD ) : Byte
1273: Function GetCValue( cmyk : COLORREF ) : Byte
1274: Function GetMValue( cmyk : COLORREF ) : Byte
1275: Function GetYValue( cmyk : COLORREF ) : Byte
1276: Function GetKValue( cmyk : COLORREF ) : Byte
1277: Function CMYK( c, m, y, k : Byte ) : COLORREF
1278: Function GetOSName: string;
1279: Function GetProcAddress( hModule : HMODULE; lpProcName : LPCSTR ) : FARPROC
1280: Function GetProcAddress(Module : HMODULE; Proc : PChar): Dword
1281: Function GetSafeCallExceptionMsg : String
1282: Function GetSaturationBitmap( Value : TBitmap ) : TBitmap
1283: Function GetSaveFilter : string
1284: Function GetSaver( Ext : string ) : TBitmapLoader
1285: Function GetScrollLockKeyState : Boolean
1286: Function GetSearchString : string
1287: Function GetSelections( Alist : TList ) : TTreeNode
1288: function GETSELTEXTBUF(BUFFER:PCHAR;BUFSIZE:INTEGER):INTEGER
1289: Function GetSendToFolder : string
1290: Function GetServer : IAppServer
1291: Function GetServerList : OleVariant
1292: Function GetShadowColor( const Color : TColor; Luminance : Integer ) : TColor
1293: Function GetShellProcessHandle : THandle
1294: Function GetShellProcessName : string
1295: Function GetShellVersion : Cardinal
1296: function getShortDayNames: string)
1297: Function GetShortHint(const hint: string): string
1298: function getShortMonthNames: string)
1299: Function GetSizeOfFile( const FileName : string ) : Int64;
1300: Function GetSizeOfFile1( Handle : THandle ) : Int64;
1301: Function GetStdHandle(nStdHandle: Integer): Integer; stdcall;
1302: Function GetStartmenuFolder : string
1303: Function GetStartupFolder : string
1304: Function GetStringProperty( Instance : TPersistent; const PropName : string ) : WideString
1305: Function GetSuccessor( cChar : char ) : TIdRegularExpressionState
1306: Function GetSwapFileSize : Integer
1307: Function GetSwapFileUsage : Integer
1308: Function GetSystemLocale : TIdCharSet
1309: Function GetSystemMetrics( nIndex : Integer ) : Integer
1310: Function GetSystemPathSH(Folder: Integer): TFilename ;
1311: Function GetTableNameFromQuery( const SQL : Widestring ) : Widestring

```

```

1312: Function GetTableNameFromSQL( const SQL : WideString ) : WideString
1313: Function GetTableNameFromSQLEx( const SQL : WideString; IdOption : IDENTIFIEROption ) : WideString
1314: Function GetTasksList( const List : TStrings ) : Boolean
1315: Function getTeamViewerID: string;
1316: Function GetTemplatesFolder : string
1317: Function GetText : PwideChar
1318: function GetText:PChar
1319: Function GetTextBuf( Buffer : PChar; BufSize : Integer ) : Integer
1320: function GETTEXTBUF(BUFFER:PCHAR;BUFSIZE:INTEGER):INTEGER
1321: Function GetTextItem( const Value : string ) : Longint
1322: function GETTEXTLEN:INTEGER
1323: Function GetThreadLocale: Longint; stdcall
1324: Function GetCurrentThreadId: LongWord; stdcall;
1325: Function GetTickCount : Cardinal
1326: Function GetTickDiff( const AOldTickCount, ANewTickCount : Cardinal ) : Cardinal
1327: Function GetTicketNr : longint
1328: Function GetTime : Cardinal
1329: Function GetTime : TDateTime
1330: Function GetTimeout : Integer
1331: Function GetTimeStr: String
1332: Function GetTimeString: String
1333: Function GetTodayFiles(startdir, amask: string): TStringlist;
1334: Function getTokenCounts : integer
1335: Function GetTotalPageFileMemory : Integer
1336: Function GetTotalPhysicalMemory : Integer
1337: Function GetTotalVirtualMemory : Integer
1338: Function GetUniqueFileName( const APath, APrefix, AExt : String ) : String
1339: Function GetUseNowForDate : Boolean
1340: Function GetUserDomainName( const CurUser : string ) : string
1341: Function GetUserName : string
1342: Function GetUserName: string;
1343: Function GetUserObjectName( hUserObject : THandle ) : string
1344: Function GetValueBitmap( Value : TBitmap ) : TBitmap
1345: Function GetValueMSec : Cardinal
1346: Function GetValueStr : String
1347: Function GetVersion: int;
1348: Function GetVersionString(FileName: string): string;
1349: Function GetVisibleNode( Index : LongInt ) : TOutlineNode
1350: Function GetVolumeFileSystem( const Drive : string ) : string
1351: Function GetVolumeName( const Drive : string ) : string
1352: Function GetVolumeSerialNumber( const Drive : string ) : string
1353: Function GetWebAppServices : IWebAppServices
1354: Function GetWebRequestHandler : IWebRequestHandler
1355: Function GetWindowCaption( Wnd : HWND ) : string
1356: Function GetWindowDC(hdwnd: HWND): HDC;
1357: Function GetWindowIcon( Wnd : HWND; LargeIcon : Boolean ) : HICON
1358: Function GetWindowRect(hwnd: HWND; arect: TRect): Boolean
1359: Function GetWindowsComputerID : string
1360: function GetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
1361: Function GetWindowsFolder : string
1362: Function GetWindowsServicePackVersion : Integer
1363: Function GetWindowsServicePackVersionString : string
1364: Function GetWindowsSystemFolder : string
1365: Function GetWindowsTempFolder : string
1366: Function GetWindowsUserID : string
1367: Function GetWindowsVersion : TWindowsVersion
1368: Function GetWindowsVersionString : string
1369: Function GmtOffsetStrToDateTime( S : string ) : TDateTime
1370: Function GMTToLocalDateTime( S : string ) : TDateTime
1371: Function GotoKey : Boolean
1372: Function GradToCycle( const Grads : Extended ) : Extended
1373: Function GradToDeg( const Grads : Extended ) : Extended
1374: Function GradToDeg( const Value : Extended ) : Extended;
1375: Function GradToDeg1( const Value : Double ) : Double;
1376: Function GradToDeg2( const Value : Single ) : Single;
1377: Function GradToRad( const Grads : Extended ) : Extended
1378: Function GradToRad( const Value : Extended ) : Extended;
1379: Function GradToRad1( const Value : Double ) : Double;
1380: Function GradToRad2( const Value : Single ) : Single;
1381: Function Gray32( const Intensity : Byte; const Alpha : Byte ) : TColor32
1382: Function GreenComponent( const Color32 : TColor32 ) : Integer
1383: function GUIDToString(const GUID: TGUID): string)
1384: Function HandleAllocated : Boolean
1385: function HandleAllocated: Boolean;
1386: Function HandleRequest : Boolean
1387: Function HandleRequest( Request : TWebRequest; Response : TWebResponse ) : Boolean
1388: Function HarmonicMean( const X : TDynFloatArray ) : Float
1389: Function HasAsParent( Value : TTTreeNode ) : Boolean
1390: Function HASCHILDDEFS : BOOLEAN
1391: Function HasCurValues : Boolean
1392: Function HasExtendCharacter( const s : UTF8String ) : Boolean
1393: Function HasFormat( Format : Word ) : Boolean
1394: Function HashValue( AStream : TStream ) : T5x4LongWordRecord;
1395: Function HashValue(AStream : TStream) : T4x4LongWordRecord
1396: Function HashValue(AStream: TStream): LongWord
1397: Function HashValue(AStream: TStream): Word
1398: Function HashValue1( AStream : TStream; const ABeginPos, AEndPos : Int64 ) : T5x4LongWordRecord;
1399: Function HashValue1(AStream : TStream): T4x4LongWordRecord
1400: Function HashValue128(const ASrc: string): T4x4LongWordRecord;

```

```

1401: Function HashValue128Stream(AStream: TStream): T4x4LongWordRecord;
1402: Function HashValue16( const ASrc : string) : Word;
1403: Function HashValue16stream( AStream : TStream) : Word;
1404: Function HashValue32( const ASrc : string) : LongWord;
1405: Function HashValue32Stream( AStream : TStream) : LongWord;
1406: Function HasMergeConflicts : Boolean
1407: Function hasMoreTokens : boolean
1408: Function HASPARENT : BOOLEAN
1409: function HasParent: Boolean
1410: Function HasTransaction( Transaction : TDBXTransaction) : Boolean
1411: Function HasUTF8BOM( S : TStream) : boolean;
1412: Function HasUTF8BOM1( S : AnsiString) : boolean;
1413: Function Haversine( X : Float) : Float
1414: Function Head( s : string; const subs : string; var tail : string) : string
1415: function HELPCOMMAND(COMMAND:INTEGER;DATA:LONGINT):BOOLEAN
1416: function HELPCONTEXT(CONTEXT:THELPCONTEXT):BOOLEAN
1417: function HELPJUMP(JUMPID:STRING):BOOLEAN
1418: Function HeronianMean( const a, b : Float) : Float
1419: function HexStrToStr(Value: string): string;
1420: function HexToBin(Text,Buffer:PChar; BufSize:Integer):Integer;
1421: function HexToBin2(HexNum: string): string;
1422: Function Hex.ToDouble( const Hex : string) : Double
1423: function HexToInt(hexnum: string): LongInt;
1424: function HexToStr(Value: string): string;
1425: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
1426: function Hi(vdat: word): byte;
1427: function HiByte(W: Word): Byte
1428: function High: Int64;
1429: Function HighlightCell(DataCol,DataRow: Integer; const Value:string; AState:TGridDrawState): Boolean
1430: function HINSTANCE: longword;
1431: function HiWord(l: DWORD): Word
1432: function HMODULE: longword;
1433: Function HourOf( const AValue : TDateTime) : Word
1434: Function HourOfTheDay( const AValue : TDateTime) : Word
1435: Function HourOfTheMonth( const AValue : TDateTime) : Word
1436: Function HourOfTheWeek( const AValue : TDateTime) : Word
1437: Function HourOfTheYear( const AValue : TDateTime) : Word
1438: Function HoursBetween( const ANow, AThen : TDateTime) : Int64
1439: Function HourSpan( const ANow, AThen : TDateTime) : Double
1440: Function HLSLTORGB1( const H, S, L : Single) : TColor32;
1441: Function HTMLDecode( const AStr : String) : String
1442: Function HTMLEncode( const AStr : String) : String
1443: Function HTMLEscape( const Str : string) : string
1444: Function HtmlTable( DataSet : TDataSet; DataSetHandler : TDSTableProducer; MaxRows : Integer) : string
1445: Function HTTPDecode( const AStr : String) : string
1446: Function HTTPPEncode( const AStr : String) : string
1447: Function Hypot( const X, Y : Extended) : Extended
1448: Function IBMax( n1, n2 : Integer) : Integer
1449: Function IBMIn( n1, n2 : Integer) : Integer
1450: Function IBRandomString( iLength : Integer) : String
1451: Function IBRandomInteger( iLow, iHigh : Integer) : Integer
1452: Function IBStripString( st : String; CharsToStrip : String) : String
1453: Function IBFormatIdentifier( Dialect : Integer; Value : String) : String
1454: Function IBFormatIdentifierValue( Dialect : Integer; Value : String) : String
1455: Function IBExtractIdentifier( Dialect : Integer; Value : String) : String
1456: Function IBQuoteIdentifier( Dialect : Integer; Value : String) : String
1457: Function IBAddIBParamSQLForDetail(Params:TParams;SQL:string;Native:Boolean;Dialect:Integer):string
1458: Procedure IBDecomposeDatabaseName(DatabaseName:String;var ServerName,Protocol,DatabasePath:String)
1459: Function RandomString( iLength : Integer) : String';
1460: Function RandomInteger( iLow, iHigh : Integer) : Integer';
1461: Function StripString( st : String; CharsToStrip : String) : String';
1462: Function FormatIdentifier( Dialect : Integer; Value : String) : String';
1463: Function FormatIdentifierValue( Dialect : Integer; Value : String) : String';
1464: Function ExtractIdentifier( Dialect : Integer; Value : String) : String';
1465: Function QuoteIdentifier( Dialect : Integer; Value : String) : String';
1466: Function AddIBParamSQLForDetail(Params:TParams;SQL:string;Native:Boolean;Dialect:Integer):string;
1467: Procedure DecomposeDatabaseName(DatabaseName:String;var ServerName,Protocol,DatabasePath:String);
1468: function NextSQLToken(var p: PChar; var Token: String; CurSection: TSQLToken): TSQLToken;
1469: Function IconToBitmap( Ico : HICON) : TBitmap
1470: Function IconToBitmap2( Ico : HICON; Size : Integer; TransparentColor : TColor) : TBitmap
1471: Function IconToBitmap3( Ico : HICON; Size : Integer; TransparentColor : TColor) : TBitmap
1472: function IdentToCharset(const Ident: string; var Charset: Longint): Boolean
1473: function IdentToColor(const Ident: string; var Color: Longint): Boolean
1474: function IdentToCursor(const Ident: string; var cursor: Longint): Boolean;
1475: Function IdGetDefault CharSet : TIdCharSet
1476: function IDispatchInvoke(Self:IDispatch;ProperSet:Boolean;const Name:String;Par:array of variant):variant
1477: Function IdPorts2 : TStringList
1478: Function IdToMib( const Id : string) : string
1479: Function IdSHA1Hash(apath: string): string;
1480: Function IdHashSHA1(apath: string): string;
1481: Function IfStr( const bCondition : boolean; const sTrue : string; const sFalse : string) : string
1482: Function IfThen( AValue : Boolean; const ATrue : string; AFalse : string) : string;
1483: Function IfThenInt( AValue : Boolean; const ATrue : integer; AFALSE : integer): integer;;
1484: Function IfThenDouble( AValue : Boolean; const ATrue : double; AFALSE : double): double;;
1485: Function IfThenBool( AValue : Boolean; const ATrue : boolean; AFALSE : boolean): boolean;;
1486: Function iif1( ATest : Boolean; const ATrue : Integer; const AFALSE : Integer) : Integer;
1487: Function iif2( ATest : Boolean; const ATrue : string; const AFALSE : string) : string;
1488: Function iif3( ATest : Boolean; const ATrue : Boolean; const AFALSE : Boolean) : Boolean;
1489: function ImportTest(S1: string;s2:longint; s3:Byte; s4:word; var s5:string): string;

```

```

1490: Function IncDay( const AValue : TDateTime; const ANumberOfDays : Integer ) : TDateTime
1491: Function IncHour( const AValue : TDateTime; const ANumberOfHours : Int64 ) : TDateTime
1492: Function IncLimit( var B : Byte; const Limit : Byte; const Incr : Byte ) : Byte;
1493: Function IncLimit1( var B : Shortint; const Limit : Shortint; const Incr : Shortint ) : Shortint;
1494: Function IncLimit2( var B : Smallint; const Limit : Smallint; const Incr : Smallint ) : Smallint;
1495: Function IncLimit3( var B : Word; const Limit : Word; const Incr : Word ) : Word;
1496: Function IncLimit4( var B : Integer; const Limit : Integer; const Incr : Integer ) : Integer;
1497: Function IncLimit5( var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal ) : Cardinal;
1498: Function IncLimit6( var B : Int64; const Limit : Int64; const Incr : Int64 ) : Int64;
1499: Function IncLimitClamp( var B : Byte; const Limit : Byte; const Incr : Byte ) : Byte;
1500: Function IncLimitClamp1( var B : Shortint; const Limit : Shortint; const Incr : Shortint ) : Shortint;
1501: Function IncLimitClamp2( var B : Smallint; const Limit : Smallint; const Incr : Smallint ) : Smallint;
1502: Function IncLimitClamp3( var B : Word; const Limit : Word; const Incr : Word ) : Word;
1503: Function IncLimitClamp4( var B : Integer; const Limit : Integer; const Incr : Integer ) : Integer;
1504: Function IncLimitClamp5( var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal ) : Cardinal;
1505: Function IncLimitClamp6( var B : Int64; const Limit : Int64; const Incr : Int64 ) : Int64;
1506: Function IncludeTrailingBackslash( S : string ) : string
1507: function IncludeTrailingBackslash(const S: string): string
1508: Function IncludeTrailingPathDelimiter( const APath : string ) : string
1509: Function IncludeTrailingPathDelimiter( S : string ) : string
1510: function IncludeTrailingPathDelimiter(const S: string): string
1511: Function IncludeTrailingSlash( const APath : string ) : string
1512: Function IncMillisecond( const AValue : TDateTime; const ANumberOfMilliseconds : Int64 ) : TDateTime
1513: Function IncMinute( const AValue : TDateTime; const ANumberOfMinutes : Int64 ) : TDateTime
1514: Function IncMonth( DateTime : TDateTime; NumberOfMonths : Integer ) : TDateTime
1515: function IncMonth(const DateTime: TDateTime; NumberOfMonths: Integer): TDateTime
1516: Function IncSecond( const AValue : TDateTime; const ANumberOfSeconds : Int64 ) : TDateTime
1517: Function IncWeek( const AValue : TDateTime; const ANumberOfWeeks : Integer ) : TDateTime
1518: Function IncYear( const AValue : TDateTime; const ANumberOfYears : Integer ) : TDateTime
1519: Function IndexOf( AClass : TClass ) : Integer
1520: Function IndexOf( AComponent : TComponent ) : Integer
1521: Function IndexOf( AObject : TObject ) : Integer
1522: Function INDEXOF( const ANAME : String ) : INTEGER
1523: Function IndexOf( const DisplayName : string ) : Integer
1524: Function IndexOf( const Item : TBookmarkStr ) : Integer
1525: Function IndexOf( const S : WideString ) : Integer
1526: Function IndexOf( const View : TJclFileMappingView ) : Integer
1527: Function INDEXOF( FIELD : TFIELD ) : INTEGER
1528: Function IndexOf( ID : LCID ) : Integer
1529: Function INDEXOF( ITEM : TMENUITEM ) : INTEGER
1530: Function IndexOf( Value : TListItem ) : Integer
1531: Function IndexOf( Value : TTreeNode ) : Integer
1532: function IndexOf(const S: string): Integer
1533: Function IndexOfName( const Name : WideString ) : Integer
1534: function IndexOfName(Name: string): Integer
1535: Function IndexOfObject( AObject : TObject ) : Integer
1536: function IndexofObject(AObject:tObject):Integer
1537: Function IndexOfTabAt( X, Y : Integer ) : Integer
1538: Function IndexStr( const AText : string; const AValues : array of string ) : Integer
1539: Function IndexText( const AText : string; const AValues : array of string ) : Integer
1540: Function IndexOfInteger( AList : TStringList; Value : Variant ) : Integer
1541: Function IndexOfFloat( AList : TStringList; Value : Variant ) : Integer
1542: Function IndexOfDate( AList : TStringList; Value : Variant ) : Integer
1543: Function IndexOfString( Alist : TStringList; Value : Variant ) : Integer
1544: Function IndyCompareStr( const A1 : string; const A2 : string ) : Integer
1545: Function IndyGetHostName : string
1546: Function IndyInterlockedDecrement( var I : Integer ) : Integer
1547: Function IndyInterlockedExchange( var A : Integer; B : Integer ) : Integer
1548: Function IndyInterlockedExchangeAdd( var A : Integer; B : Integer ) : Integer
1549: Function IndyInterlockedIncrement( var I : Integer ) : Integer
1550: Function IndyLowerCase( const A1 : string ) : string
1551: Function IndyStrToBool( const AString : String ) : Boolean
1552: Function IndyUpperCase( const A1 : string ) : string
1553: Function InitCommonControl( CC : Integer ) : Boolean
1554: Function InitTempPath : string
1555: Function InMainThread : boolean
1556: Function inOpArray( W : WideChar; sets : array of WideString ) : boolean
1557: Function Input : Text
1558: Function InputBox( const ACaption, APrompt, ADefault : string ) : string
1559: function InputBox(const ACaption: string; const APrompt: string; const ADefault: string): string
1560: Function InputLn(const AMask: string; AEcho:Boolean;ATabWidth:Integer;AMaxLineLength:Integer): string
1561: Function InputQuery( const ACaption, APrompt : string; var Value : string ) : Boolean
1562: function InputQuery(const ACaption: string; const APrompt: string; var Value: string): Boolean
1563: Function InquireSignal( RtlSignum : Integer ) : TSignalState
1564: Function InRanger( const A, Min, Max : Double ) : Boolean
1565: function Insert( Index : Integer ) : TCollectionItem
1566: Function Insert( Index : Integer ) : TComboExItem
1567: Function Insert( Index : Integer ) : THeaderSection
1568: Function Insert( Index : Integer ) : TListItem
1569: Function Insert( Index : Integer ) : TStatusPanel
1570: Function Insert( Index : Integer ) : TWorkArea
1571: Function Insert( Index : LongInt; const Text : string ) : LongInt
1572: Function Insert( Sibling : TTreeNode; const S : string ) : TTreeNode
1573: Function INSERTNEWLINEAFTER( AITEM : TMENUITEM ) : INTEGER
1574: Function INSERTNEWLINEBEFORE( AITEM : TMENUITEM ) : INTEGER
1575: Function InsertNode( Node, Sibling : TTreeNode; const S : string; Ptr : Pointer ) : TTreeNode
1576: Function InsertObject( Index : LongInt; const Text : string; const Data : Pointer ) : LongInt
1577: Function InsertObject( Sibling : TTreeNode; const S : string; Ptr : Pointer ) : TTreeNode
1578: Function Instance : Longint

```

```

1579: function InstanceSize: Longint
1580: Function Int(e : Extended) : Extended;
1581: function Int64ToStr(i: Int64): String;
1582: Function IntegerToBcd( const AValue : Integer) : TBcd
1583: Function Intensity( const Color32 : TColor32) : Integer;
1584: Function Intensity( const R, G, B : Single) : Single;
1585: Function InterestPayment(const Rate:Extended;Period,NPeriods:Integer;const PresentValue,
FutureValue:Extended; PaymentTime : TPaymentTime) : Extended
1586: Function InterestRate(NPeriods:Integer;const Payment,PresVal,
FutureVal:Extended;PaymentTime:TPaymentTime):Extended
1587: Function InternalDecodeDate( DateTime : TDateTime; var Year, Month, Day, DOW : Word) : Boolean
1588: Function InternalRateOfReturn( const Guess : Extended; const CashFlows : array of Double) : Extended
1589: Function InternalUpdateRecord( Tree : TUpdateTree) : Boolean
1590: Function IntersectRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
1591: function IntersectRect(out Rect: TRect; const R1, R2: TRect): Boolean)
1592: Function IntMibToStr( const Value : string) : string
1593: Function IntPower( const Base : Extended; const Exponent : Integer) : Extended
1594: Function IntToBin( Value : cardinal) : string
1595: Function IntToHex( Value : Integer; Digits : Integer) : string;
1596: function IntToHex(a: integer; b: integer): string;
1597: Function IntToHex64( Value : Int64; Digits : Integer) : string;
1598: function IntToHex64(Value: Int64; Digits: Integer): string)
1599: Function IntTo3Str( Value : Longint; separator: string) : string
1600: Function inttobool( aInt : LongInt) : Boolean
1601: function IntToStr(i: Int64): String;
1602: Function IntToStr64(Value: Int64): string)
1603: function IOResult: Integer
1604: Function IPv6AddressToStr(const AValue: TIIPv6Address) : string
1605: Function IsAccel(VK: Word; const Str: string): Boolean
1606: Function IsAddressInNetwork( Address : String) : Boolean
1607: Function IsAdministrator : Boolean
1608: Function IsAlias( const Name : string) : Boolean
1609: Function IsApplicationRunning( const AClassName, ApplName : string) : Boolean
1610: Function IsASCII( const AByte : Byte) : Boolean;
1611: Function IsASCIILDH( const AByte : Byte) : Boolean;
1612: Function IsAssembly(const FileName: string): Boolean;
1613: Function IsBcdNegative( const Bcd : TBcd) : Boolean
1614: Function IsBinary(const AChar : Char) : Boolean
1615: function IsConsole: Boolean)
1616: Function IsDelimiter( Delimiters, S : string; Index : Integer) : Boolean
1617: function IsDelimiter(const Delimiters: string; const S: string; Index: Integer): Boolean)
1618: Function IsDelphiDesignMode : boolean
1619: Function IsDelphiRunning : boolean
1620: Function IsDFAState : boolean
1621: Function IsDirectory( const FileName : string) : Boolean
1622: Function IsDomain( const S : String) : Boolean
1623: function IsDragObject(Sender: TObject): Boolean;
1624: Function IsEditing : Boolean
1625: Function ISEMPTY : BOOLEAN
1626: Function IsEqual( Value : TParameters) : Boolean
1627: Function ISEQUAL( VALUE : TPARAMS) : BOOLEAN
1628: function IsEqualGUID(const guid1, guid2: TGUID): Boolean)
1629: Function IsFirstNode : Boolean
1630: Function IsFloatAtZero( const X : Float) : Boolean
1631: Function IsFormatRegistered( Extension, AppID : string) : Boolean
1632: Function IsFormOpen(const FormName: string): Boolean;
1633: Function IsFQDN( const S : String) : Boolean
1634: Function IsGrayScale : Boolean
1635: Function IsHex(AChar : Char) : Boolean;
1636: Function IsHexString(const AString: string): Boolean;
1637: Function IsHostname( const S : String) : Boolean
1638: Function IsInfinite( const AValue : Double) : Boolean
1639: Function IsInLeapYear( const AValue : TDateTime) : Boolean
1640: Function IsInternet: boolean;
1641: Function IsLeadChar( ACh : Char) : Boolean
1642: Function IsLeapYear( Year : Word) : Boolean
1643: function IsLeapYear(Year: Word): Boolean)
1644: function IsLibrary: Boolean)
1645: Function ISLINE : BOOLEAN
1646: Function IsLinkedTo( DataSet : TDataSet) : Boolean
1647: Function ISLINKEDTO( DATASOURCE : TDATASOURCE) : BOOLEAN
1648: Function IsLiteralChar( const EditMask : string; Offset : Integer) : Boolean
1649: Function IsMatch( const Pattern, Text : string) : Boolean //Grep like RegEx
1650: Function IsMainAppWindow( Wnd : HWND) : Boolean
1651: Function IsMediaPresentInDrive( Drive : Char) : Boolean
1652: function IsMemoryManagerSet: Boolean)
1653: Function IsMultiTableQuery( const SQL : WideString) : Boolean
1654: function IsMultiThread: Boolean)
1655: Function IsNumeric( AChar : Char) : Boolean;
1656: Function IsNumeric2( const AString : string): Boolean;
1657: Function IsNTFS: Boolean;
1658: Function IsOctal( AChar : Char) : Boolean;
1659: Function IsOctalString(const AString: string): Boolean;
1660: Function IsPathDelimiter( S : string; Index : Integer) : Boolean
1661: function IsPathDelimiter(const S: string; Index: Integer): Boolean)
1662: Function ISPM( const AValue : TDateTime) : Boolean
1663: Function IsPositiveFloatArray( const X : TDynFloatArray) : Boolean
1664: Function IsPortAvailable( ComNum : Cardinal) : Boolean');
1665: Function IsPrimeFactor( const F, N : Cardinal) : Boolean

```

```

1666: Function IsPrimeRM( N : Cardinal ) : Boolean //rabin miller
1667: Function IsPrimeTD( N : Cardinal ) : Boolean //trial division
1668: Function IsPrivilegeEnabled( const Privilege : string ) : Boolean
1669: Function ISqrt( const I : Smallint ) : Smallint
1670: Function IsReadOnly( const Filename: string): boolean;
1671: Function IsRectEmpty( const Rect : TRect ) : Boolean
1672: function IsRectEmpty(const Rect: TRect): Boolean)
1673: Function IsRelativePrime( const X, Y : Cardinal ) : Boolean
1674: Function ISRIGHTTOLEFT : BOOLEAN
1675: function IsRightToLeft: Boolean
1676: Function IsSameDay( const AValue, ABasis : TDateTime ) : Boolean
1677: Function ISSEQUENCED : BOOLEAN
1678: Function IsSystemModule( const Module : HMODULE ) : Boolean
1679: Function IsSystemResourcesMeterPresent : Boolean
1680: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: string): boolean;
1681: Function IsToday( const AValue : TdateTime ) : Boolean
1682: function IsToday(const AValue: TDateTime): Boolean;
1683: Function IsTopDomain( const AStr : string ) : Boolean
1684: Function IsUTF8LeadByte( Lead : Char ) : Boolean
1685: Function IsUTF8String( const s : UTF8String ) : Boolean
1686: Function IsUTF8TrailByte( Lead : Char ) : Boolean
1687: Function ISVALIDCHAR( INPUTCHAR : CHAR ) : BOOLEAN
1688: Function IsValidDate( const AYear, AMonth, ADay : Word ) : Boolean
1689: Function IsValidDateDay( const AYear, ADayOfYear : Word ) : Boolean
1690: Function IsValidDateMonthWeek( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word ) : Boolean
1691: Function IsValidDateTime(const AYear,AMonth,ADay,AHour,AMinute, ASecond, AMilliSecond: Word): Boolean
1692: Function IsValidDateWeek( const AYear, AWeekOfYear, ADayOfWeek : Word ) : Boolean
1693: Function IsValidIdent( Ident : string ) : Boolean
1694: function IsValidIdent(const Ident: string; AllowDots: Boolean): Boolean)
1695: Function IsValidIP( const S : String ) : Boolean
1696: Function IsValidTime( const AHour, AMinute, ASecond, AMilliSecond : Word ) : Boolean
1697: Function IsValidISBN( const ISBN : AnsiString ) : Boolean
1698: Function IsVariantManagerSet: Boolean; //deprecated;
1699: Function IsVirtualPcGuest : Boolean;
1700: Function IsVmWareGuest : Boolean;
1701: Function IsVCLControl(Handle: HWnd): Boolean;
1702: Function IsWhiteString( const AStr : String ) : Boolean
1703: Function IsWindowResponding( Wnd : HWND; Timeout : Integer ) : Boolean
1704: Function IsWoW64: boolean;
1705: Function IsWin64: boolean;
1706: Function IsWow64String(var s: string): Boolean;
1707: Function IsWin64String(var s: string): Boolean;
1708: Function IsWindowsVista: boolean;
1709: Function isPowerof2(num: int64): boolean;
1710: Function powerOf2(exponent: integer): int64;
1711: function IsZero(const A: Extended; Epsilon: Extended): Boolean //overload;
1712: function IsZero1(const A: Double; Epsilon: Double): Boolean //overload;
1713: function IsZero2(const A: Single; Epsilon: Single): Boolean //overload;
1714: Function ItemAtPos( Pos : TPoint; IgnoreTabHeight : Boolean ) : Integer
1715: function ITEMATPOS(POS:TPOINT:EXISTING:BOOLEAN):INTEGER
1716: Function ItemRect( Index : Integer ) : TRect
1717: function ITEMRECT(INDEX:INTEGER):TRECT
1718: Function ItemWidth( Index : Integer ) : Integer
1719: Function JavahashCode(val: string): Integer;
1720: Function JosephusG(n,k: integer; var graphout: string): integer;
1721: Function JulianDateToDate( const AValue : Double ) : TDateTime
1722: Function JustName(PathName : string) : string; //in path and ext
1723: Function JvMessageBox( const Text, Caption : string; Flags : DWORD ) : Integer;
1724: Function JvMessageBox1( const Text : string; Flags : DWORD ) : Integer;
1725: Function Keepalive : Boolean
1726: Function KeysToShiftState(Keys: Word): TShiftState;
1727: Function KeyDataToShiftState(KeyData: Longint): TShiftState;
1728: Function KeyboardStateToShiftState2(const KeyboardState: TKeyboardState): TShiftState;
1729: Function KeyboardStateToShiftState: TShiftState; overload;
1730: Function Languages : TLanguages
1731: Function Last : TClass
1732: Function Last : TComponent
1733: Function Last : TObject
1734: Function LastDelimiter( Delimiters, S : string ) : Integer
1735: function LastDelimiter(const Delimiters: string; const S: string): Integer;
1736: Function LastPos( const ASubstr : string; const AStr : string ) : Integer
1737: Function Latitude2WGS84(lat: double): double;
1738: Function LCM(m,n:longint):longint;
1739: Function LCMJ( const X, Y : Cardinal ) : Cardinal
1740: Function Ldexp( const X : Extended; const P : Integer ) : Extended
1741: Function LeftStr( const AText : AnsiString; const ACount : Integer ) : AnsiString;
1742: Function LeftStr( const ATText : WideString; const ACount : Integer ) : WideString;
1743: function Length: Integer;
1744: Procedure LetfileList(FileList: TStringlist; apath: string);
1745: function lengthmp3(mp3path: string):integer;
1746: Function LineInRect( const P1, P2 : TPoint; const Rect : TRect ) : Boolean;
1747: Function LineInRect1( const P1, P2 : TFloatPoint; const Rect : TFloatRect ) : Boolean;
1748: Function LineSegmentIntersection(const L1P1 : TFloatPoint; L1P2 : TFloatPoint; const L2P1 : TFloatPoint;
L2P2 : TFloatPoint; var P : TFloatPoint) : Boolean;
1749: function LineStart(Buffer, BufPos: PChar): PChar
1750: function LineStart(Buffer, BufPos: PChar): PChar)
1751: function ListSeparator: char;
1752: function Ln(x: Extended): Extended;
1753: Function LnXP1( const X : Extended ) : Extended

```

```

1754: function Lo(vdat: word): byte;
1755: Function LoadCursor(hInstance: HINST; lpCursorName: PChar): HCURSOR
1756: Function LoadedModulesList( const List : TStrings; ProcessID : DWORD; HandlesOnly : Boolean ) : Boolean
1757: Function LoadFileAsString( const FileName : string ) : string
1758: Function LoadFromFile( const FileName : string ) : TBitmapLoader
1759: Function LoadLibraryEx(LibName: PChar; hFile: Longint; Flags: Longint): Longint; stdcall;
1760: Function LoadPackage(const Name: string): HMODULE
1761: Function LoadResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle ) : HGLOBAL
1762: Function LoadStr( Ident : Integer ) : string
1763: Function LoadString(Instance: Longint; IDent: Integer; Buffer: PChar; Size: Integer): Integer; stdcall;
1764: Function LoadWideStr( Ident : Integer ) : WideString
1765: Function LOCATE(const KEYFIELDS: String;const KEYVALUES:VARIANT;OPTIONS: TLOCATEOPTIONS) : BOOLEAN
1766: Function LockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint ) : HRESULT
1767: Function LockServer( fLock : LongBool ) : HRESULT
1768: Function LockVolume( const Volume : string; var Handle : THandle ) : Boolean
1769: Function Log( const X : Extended ) : Extended
1770: Function Log10( const X : Extended ) : Extended
1771: Function Log2( const X : Extended ) : Extended
1772: function LogBase10(X: Float): Float;
1773: Function LogBase2(X: Float): Float;
1774: Function LogBaseN(Base, X: Float): Float;
1775: Function LogN( const Base, X : Extended ) : Extended
1776: Function LogOffOS : Boolean
1777: Function LoginDialog( const ADatabaseName : string; var AUserName, APassword : string ) : Boolean
1778: Function LoginDialogEx(const ADatabaseName:string;var AUserName,APassword:string;NameReadOnly:Bool):Bool;
1779: Function LongDateFormat: string;
1780: function LongTimeFormat: string;
1781: Function LongWordToFourChar( AC Cardinal : LongWord ) : string
1782: Function LOOKUP(const KEYFIELDS: String; const KEYVALUES: VARIANT; const RESULTFIELDS: String): VARIANT
1783: Function LookupName( const name : string ) : TInAddr
1784: Function LookupService( const service : string ) : Integer
1785: function Low: Int64;
1786: Function LowerCase( S : string ) : string
1787: Function Lowercase(s : AnyString) : AnyString;
1788: Function LRot( const Value : Byte; const Count : TBitRange ) : Byte;
1789: Function LRot1( const Value : Word; const Count : TBitRange ) : Word;
1790: Function LRot2( const Value : Integer; const Count : TBitRange ) : Integer;
1791: function MainInstance: longword
1792: function MainThreadID: longword
1793: Function Map(x, in_min, in_max, out_min, out_max: integer): integer; //arduino
1794: Function mapMax(ax, in_min, in_max, out_min, out_max: integer): integer;
1795: Function MakeCanonicalIPv4Address( const AAAddr : string ) : string
1796: Function MakeCanonicalIPv6Address( const AAAddr : string ) : string
1797: Function MakeDIB( out Bitmap : PBitmapInfo ) : Integer
1798: Function MakeDWordIntoIPv4Address( const ADWord : Cardinal ) : string
1799: function MakeLong(A, B: Word): Longint
1800: Function MakeTempFilename( const APath : String ) : string
1801: Function MakeValidFileName( const Str : string ) : string
1802: Function MakeValueMap( Enumeration : string; ToCds : Boolean ) : string
1803: function MakeWord(A, B: Byte): Word
1804: Function MakeYear4Digit( Year, Pivot : Integer ) : Integer
1805: Function MapDateTime(const DateFormatType:string; DateFormat:string;Value:string;ToCds:Boolean): string
1806: Function MapValues( Mapping : string; Value : string ) : string
1807: Function MaskDoFormatText( const EditMask : string; const Value : string; Blank : Char ) : string
1808: Function MaskGetCharType( const EditMask : string; MaskOffset : Integer ) : TMaskCharType
1809: Function MaskGetCurrentDirectives( const EditMask : string; MaskOffset : Integer ) : TMaskDirectives
1810: Function MaskGetFldSeparator( const EditMask : string ) : Integer
1811: Function MaskGetMaskBlank( const EditMask : string ) : Char
1812: Function MaskGetMaskSave( const EditMask : string ) : Boolean
1813: Function MaskIntLiteralToChar( IChar : Char ) : Char
1814: Function MaskOffsetToOffset( const EditMask : String; MaskOffset : Integer ) : Integer
1815: Function MaskOffsetToWideOffset( const EditMask : String; MaskOffset : Integer ) : Integer
1816: Function MaskString( Mask, Value : String ) : String
1817: Function Match( const sString : string ) : TniRegularExpressionMatchResult
1818: Function Match1( const sString : string; iStart : integer ) : TniRegularExpressionMatchResult
1819: Function Matches( const Filename : string ) : Boolean
1820: Function MatchesMask( const Filename, Mask : string ) : Boolean
1821: Function MatchStr( const AText : string; const AValues : array of string ) : Boolean
1822: Function MatchText( const AText : string; const AValues : array of string ) : Boolean
1823: Function Max( AValueOne, AValueTwo : Integer ) : Integer
1824: function Max(const x,y: Integer): Integer;
1825: Function Max1( const B1, B2 : Shortint ) : Shortint;
1826: Function Max2( const B1, B2 : Smallint ) : Smallint;
1827: Function Max3( const B1, B2 : Word ) : Word;
1828: function Max3(const x,y,z: Integer): Integer;
1829: Function Max4( const B1, B2 : Integer ) : Integer;
1830: Function Max5( const B1, B2 : Cardinal ) : Cardinal;
1831: Function Max6( const B1, B2 : Int64 ) : Int64;
1832: Function Max64( const AValueOne, AValueTwo : Int64 ) : Int64
1833: Function MaxFloat( const X, Y : Float ) : Float
1834: Function MaxFloatArray( const B : TDynFloatArray ) : Float
1835: Function MaxFloatArrayIndex( const B : TDynFloatArray ) : Integer
1836: function MaxIntValue(const Data: array of Integer):Integer
1837: Function MaxJ( const B1, B2 : Byte ) : Byte;
1838: function MaxPath: string;
1839: function MaxValue(const Data: array of Double): Double
1840: Function MaxCalc( const Formula : string ) : Extended //math expression parser
1841: Procedure MaxCalcF( const Formula : string ); //out to console memo2
1842: function MD5(const fileName: string): string;

```

```

1843: Function Mean( const Data : array of Double) : Extended
1844: Function Median( const X : TDynFloatArray) : Float
1845: Function Memory : Pointer
1846: Function MemoryPos( const ASubStr : string; MemBuff : PChar; MemorySize : Integer) : Integer
1847: Function MessageBox(hndl: cardinal; text, caption: string; utype: cardinal): Integer;
1848: function MESSAGEBOX(TEXT,CAPTION:PCCHAR;FLAGS:WORD):INTEGER
1849: Function MessageDlg(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
1850: Function MessageDlg1(const
1851:   Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;DefaultButton:TMsgDlgBtn): Integer;
1851: Function MessageDlgPos(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,
1851: Y:Integer):Integer;
1852: Function MessageDlgPos1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
1852: Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn ) : Integer;
1853: Function MessageDlgPosHelp( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
1853: Longint; X, Y : Integer; const HelpFileName : string ) : Integer;
1854: Function MessageDlgPosHelp1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
1854: Longint; X, Y : Integer; const HelpFileName : string; DefaultButton : TMsgDlgBtn ) : Integer;
1855: Function MibToId( Mib : string ) : string
1856: Function MidStr( const AText : AnsiString; const AStart, ACount : Integer ) : AnsiString;
1857: Function MidStr( const AText : WideString; const AStart, ACount : Integer ) : WideString;
1858: Function microsecondsToCentimeters(mseconds: longint): longint; //340m/s speed of sound
1859: Function Micros(const Timer:THPTimer;const TimerRunning:Boolean):Int64//TypeS('THPTimer', 'Int64
1860: Function MIDIOut( DeviceID : Cardinal ) : IJclMIDIOut
1861: Procedure GetMidiOutputs( const List : TStrings )
1862: Function MIDISingleNoteTuningData( Key : TMIDINote; Frequency : Single ) : TSingleNoteTuningData
1863: Function MIDINoteToStr( Note : TMIDINote ) : string
1864: Function WinMidiOut( DeviceID : Cardinal ) : IJclWinMidiOut
1865: Procedure GetMidiOutputs( const List : TStrings )
1866: Procedure MidiOutCheck( Code : MMResult )
1867: Procedure MidiInCheck( Code : MMResult )
1868: Function MillisecondOf( const AValue : TDateTime ) : Word
1869: Function MillisecondOfTheDay( const AValue : TDateTime ) : LongWord
1870: Function MillisecondOfTheHour( const AValue : TDateTime ) : LongWord
1871: Function MillisecondOfTheMinute( const AValue : TDateTime ) : LongWord
1872: Function MillisecondOfTheMonth( const AValue : TDateTime ) : LongWord
1873: Function MillisecondOfTheSecond( const AValue : TDateTime ) : Word
1874: Function MillisecondOfTheWeek( const AValue : TDateTime ) : LongWord
1875: Function MillisecondOfTheYear( const AValue : TDateTime ) : Int64
1876: Function MillisecondsBetween( const ANow, AThen : TDateTime ) : Int64
1877: Function MillisecondSpan( const ANow, AThen : TDateTime ) : Double
1878: Function milliToDateTIme( Millisecond : LongInt ) : TDateTime';
1879: Function Micros( const Timer : THPTimer; const TimerRunning : Boolean ) : Int64
1880: Function millis: int64;
1881: Function Min( AValueOne, AValueTwo : Integer ) : Integer
1882: Function Min1( const B1, B2 : Shortint ) : Shortint;
1883: Function Min2( const B1, B2 : Smallint ) : Smallint;
1884: Function Min3( const B1, B2 : Word ) : Word;
1885: Function Min4( const B1, B2 : Integer ) : Integer;
1886: Function Min5( const B1, B2 : Cardinal ) : Cardinal;
1887: Function Min6( const B1, B2 : Int64 ) : Int64;
1888: Function Min64( const AValueOne, AValueTwo : Int64 ) : Int64
1889: Function MinClientRect : TRect;
1890: Function MinClientRect1( IncludeScroller : Boolean ) : TRect;
1891: Function MinClientRect2( TabCount : Integer; IncludeScroller : Boolean ) : TRect;
1892: Function MinFloat( const X, Y : Float ) : Float
1893: Function MinFloatArray( const B : TDynFloatArray ) : Float
1894: Function MinFloatArrayIndex( const B : TDynFloatArray ) : Integer
1895: Function MinimizeName( const Filename : string; Canvas : TCanvas; MaxLen : Integer ) : string
1896: Function MinimizeName( const Filename : TFileName; Canvas : TCanvas; MaxLen : Integer ) : TFileName
1897: function MinimizeName(const Filename: String; Canvas: TCanvas;MaxLen: Integer): TFileName
1898: Function MinIntValue( const Data : array of Integer ) : Integer
1899: function MinIntValue(const Data: array of Integer):Integer)
1900: Function MinJ( const B1, B2 : Byte ) : Byte;
1901: Function MinuteOf( const AValue : TDateTime ) : Word
1902: Function MinuteOfTheDay( const AValue : TDateTime ) : Word
1903: Function MinuteOfTheHour( const AValue : TDateTime ) : Word
1904: Function MinuteOfTheMonth( const AValue : TDateTime ) : Word
1905: Function MinuteOfTheWeek( const AValue : TDateTime ) : Word
1906: Function MinuteOfTheYear( const AValue : TDateTime ) : LongWord
1907: Function MinutesBetween( const ANow, AThen : TDateTime ) : Int64
1908: Function MinuteSpan( const ANow, AThen : TDateTime ) : Double
1909: Function MinValue( const Data : array of Double ) : Double
1910: function MinValue(const Data: array of Double): Double)
1911: Function MixerLeftRightToArray( Left, Right : Cardinal ) : TDynCardinalArray
1912: Function MMCheck( const MciError : MCIERROr; const Msg : string ) : MCIERROr
1913: Function ModFloat( const X, Y : Float ) : Float
1914: Function ModifiedJulianDateToDateTIme( const AValue : Double ) : TDateTime
1915: Function Modify( const Key : string; Value : Integer ) : Boolean
1916: Function ModuleCacheID : Cardinal
1917: Function ModuleFromAddr( const Addr : Pointer ) : HMODULE
1918: Function MonitorFromPoint( const Point : TPoint; MonitorDefault : TMonitorDefaultTo ) : TMonitor
1919: Function MonitorFromRect( const Rect : TRect; MonitorDefault : TMonitorDefaultTo ) : TMonitor
1920: Function MonitorFromWindow( const Handle : THandle; MonitorDefault : TMonitorDefaultTo ) : TMonitor
1921: Function MonthOf( const AValue : TDateTime ) : Word
1922: Function MonthOfTheYear( const AValue : TDateTime ) : Word
1923: Function MonthsBetween( const ANow, AThen : TDateTime ) : Integer
1924: Function MonthSpan( const ANow, AThen : TDateTime ) : Double
1925: Function MonthStr( DateTIme : TDateTime ) : string
1926: Function MouseCoord( X, Y : Integer ) : TGridCoord

```

```

1927: Function MOVEBY( DISTANCE : INTEGER ) : INTEGER
1928: Function MoveFile( Source, Dest : string; Flags : FILEOP_FLAGS ) : Boolean
1929: Function MoveNext : Boolean
1930: Function MSecsToTimeStamp( MSecs : Comp ) : TTimeStamp
1931: function MSecsToTimeStamp(MSecs: Comp): TTimeStamp
1932: Function Name : string
1933: Function NetPresentValue(const Rate:Extended;const CashFlows:array of
Double;PaymentTime:TPaymentTime):Extended
1934: function NetworkVolume(DriveChar: Char): string
1935: Function NEWBOTOMLINE : INTEGER
1936: Function NewCompareNode( Field : TField; Operator : TCANOperator; const Value : Variant ) : PExprNode
1937: Function NEWITEM( const ACAPTION : STRING; ASHORTCUT : TSHORCUT; ACHECKED, AENABLED : BOOLEAN; AONCLICK : TNTOFIEVENT; HCTX : WORD; const ANAME : String ) : TMENUITEM
1938: Function NEWLINE : TMENUITEM
1939: Function NEWMENU( OWNER : TCOMPONENT; const ANAME : STRING; ITEMS : array of TMenuItem ) : TMAINMENU
1940: Function NewNode(Kind: TExprNodeKind; Operator:TCANOperator; const Data:Variant; Left,
Right:PExprNode):PExprNode
1941: Function NEWPOPPUPMENU(OWNER:TCOMPONENT;const ANAME:String; ALIGNMENT:TPOPUPALIGNMENT;AUTOPOPUP:BOOLEAN;
const ITEMS : array of TCMENUITEM) : TPOPUPMENU
1942: Function NewState( eType : TniRegularExpressionStateType ) : TniRegularExpressionState
1943: Function NEWSUBMENU(const ACAPT:String;HCTX:WORD;const ANAME:String;ITEMS:array of
TMenuItem;AENABLED:BOOL): TMENUITEM
1944: Function NEWTOPLINE : INTEGER
1945: Function Next : TIAuthWhatsNext
1946: Function NextCharIndex( S : String; Index : Integer ) : Integer
1947: Function NextRecordSet : TCustomSQLDataSet
1948: Function NextRecordset( var RecordsAffected : Integer ) : _Recordset
1949: Function NextSQLToken1( var p : WideChar; out Token : WideString; CurSection : TSQLOken ) : TSQLOken;
1950: Function NextToken : Char
1951: Function nextToken : WideString
1952: function NextToken:Char
1953: Function Norm( const Data : array of Double ) : Extended
1954: Function NormalizeAngle( const Angle : Extended ) : Extended
1955: Function NormalizeBcd( const InBcd : TBcd; var OutBcd : TBcd; const Prec, Scale : Word ) : Boolean
1956: Function NormalizeRect( const Rect : TRect ) : TRect
1957: function NormalizeRect(const Rect: TRect): TRect;
1958: Function Now : TDateTime
1959: function Now2: tDateTime
1960: Function NumProcessThreads : integer
1961: Function NumThreadCount : integer
1962: Function NthDayOfWeek( const AValue : TDateTime ) : Word
1963: Function NtProductType : TNTProductType
1964: Function NtProductTypeString : string
1965: function Null: Variant;
1966: Function NullPoint : TPoint
1967: Function NullRect : TRect
1968: Function Null2Blank(aString:String):String;
1969: Function NumberOfPeriods( const Rate : Extended; Payment : Extended; const PresentValue, FutureValue : Extended;
PaymentTime : TPaymentTime ) : Extended
1970: Function NumIP : integer
1971: function Odd(x: Longint): boolean;
1972: Function OffsetFromUTC : TDateTime
1973: Function OffsetPoint( const P, Offset : TPoint ) : TPoint
1974: Function OffsetRect( var Rect : TRect; DX : Integer; DY : Integer ) : Boolean
1975: function OffsetRect(var Rect: TRect; DX:Integer; DY:Integer): Boolean
1976: Function OffsetToMaskOffset( const EditMask : string; Offset : Integer ) : Integer
1977: Function OkToChangeFieldAlignment( AField : TField; Alignment : TAlignment ) : Boolean
1978: Function OldBCDToCurr( const BCD : TBcd; var Curr : Currency ) : Boolean
1979: Function OldCurrToBCD(const Curr:Currency; var BCD:TBcd; Precision:Integer;Decimals:Integer): Boolean
1980: function OpenBit:Integer
1981: Function OpenDatabase : TDatabase
1982: Function OpenDatabase( const DatabaseName : string ) : TDatabase
1983: Procedure OpenDir(adir: string);
1984: Function OpenGLColorToWinColor( const Red, Green, Blue : Float ) : TColor
1985: Function OpenObject( Value : PChar ) : Boolean
1986: Function OpenObject1( Value : string ) : Boolean;
1987: Function OpenSession( const SessionName : string ) : TSession
1988: Function OpenVolume( const Drive : Char ) : THandle
1989: function OrdFourByteToCardinal(AByte1, AByte2, AByte3, AByte4 : Byte) : Cardinal
1990: Function OrdFourByteToLongWord( AByte1, AByte2, AByte3, AByte4 : Byte ) : LongWord
1991: Function OrdToBinary( const Value : Byte) : string;
1992: Function OrdToBinary1( const Value : Shortint ) : string;
1993: Function OrdToBinary2( const Value : Smallint ) : string;
1994: Function OrdToBinary3( const Value : Word ) : string;
1995: Function OrdToBinary4( const Value : Integer ) : string;
1996: Function OrdToBinary5( const Value : Cardinal ) : string;
1997: Function OrdToBinary6( const Value : Int64 ) : string;
1998: Function OSCheck( RetVal : Boolean ) : Boolean
1999: Function OSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD ) : string
2000: Function OSIdentToString( const OSIdent : DWORD ) : string
2001: Function Output: Text
2002: Function Overlay( ImageIndex : Integer; Overlay : TOverlay ) : Boolean
2003: Function Owner : TCustomListView
2004: function Owner : TPersistent
2005: Function PadInputLiterals( const EditMask : String; const Value : string; Blank : Char ) : string
2006: Function PadL( pStr : String; pLth : integer ) : String
2007: Function Padl(s : AnyString;I : longInt) : AnyString;
2008: Function PadLCh( pStr : String; pLth : integer; pChr : char ) : String
2009: Function PadR( pStr : String; pLth : integer ) : String

```

```

2010: Function Padr(s : AnyString;I : longInt) : AnyString;
2011: Function PadRCh( pStr : String; pLth : integer; pChr : char) : String
2012: Function PadString( const AString : String; const ALen : Integer; const AChar : Char) : String
2013: Function Padz(s : AnyString;I : longInt) : AnyString;
2014: Function PaethPredictor( a, b, c : Byte) : Byte
2015: Function PARAMBYNAME( const VALUE : String) : TPARAM
2016: Function ParamByName( const Value : WideString) : TParameter
2017: Function ParamCount: Integer
2018: Function ParamsEncode( const ASrc : string) : string
2019: function ParamStr(Index: Integer): string
2020: Function ParseDate( const DateStr : string) : TDateTime
2021: Function PARSESQL( SQL : String; DCREATE : BOOLEAN) : String
2022: Function ParseSQL( SQL : WideString; DoCreate : Boolean) : WideString
2023: Function PathAddExtension( const Path, Extension : string) : string
2024: Function PathAddSeparator( const Path : string) : string
2025: Function PathAppend( const Path, Append : string) : string
2026: Function PathBuildRoot( const Drive : Byte) : string
2027: Function PathCanonicalize( const Path : string) : string
2028: Function PathCommonPrefix( const Path1, Path2 : string) : Integer
2029: Function PathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string;
2030: Function PathCompactPath1(const Canv:TCanvas;const Path:string;const Width:Int;CmpFmt:TCompactPath):string;
2031: Function PathEncode( const ASrc : string) : string
2032: Function PathExtractFileDirFixed( const S : AnsiString) : AnsiString
2033: Function PathExtractFileNameNoExt( const Path : string) : string
2034: Function PathExtractPathDepth( const Path : string; Depth : Integer) : string
2035: Function PathGetDepth( const Path : string) : Integer
2036: Function PathGetLongName( const Path : string) : string
2037: Function PathGetLongName2( Path : string) : string
2038: Function PathGetShortName( const Path : string) : string
2039: Function PathIsAbsolute( const Path : string) : Boolean
2040: Function PathIsChild( const Path, Base : AnsiString) : Boolean
2041: Function PathIsDiskDevice( const Path : string) : Boolean
2042: Function PathIsUNC( const Path : string) : Boolean
2043: Function PathRemoveExtension( const Path : string) : string
2044: Function PathRemoveSeparator( const Path : string) : string
2045: Function Payment(Rate:Extended;NPeriods:Int;const PresentVal,
FutureVal:Extended;PaymentTime:TPaymentTime):Extended
2046: Function Peek : Pointer
2047: Function Peek : TObject
2048: function PERFORM(MSG:CARDINAL;WPARAM:LPARAM;LONGINT):LONGINT
2049: Function PeriodPayment(const Rate:Extended;Period,NPeriods:Integer; const PresentValue, FutureValue :
Extended; PaymentTime : TPaymentTime) : Extended
2050: function Permutation(npr, k: integer): extended;
2051: function PermutationInt(npr, k: integer): Int64;
2052: Function PermutationJ( N, R : Cardinal) : Float
2053: Function Pi : Extended;
2054: Function PiE : Extended;
2055: Function PixelsToDialogsX( const Pixels : Word) : Word
2056: Function PixelsToDialogsY( const Pixels : Word) : Word
2057: Function PlaySound(s: pchar; flag, syncflag: integer): boolean;
2058: Function Point( X, Y : Integer) : TPoint
2059: function Point(X, Y: Integer): TPoint
2060: Function PointAssign( const X, Y : Integer) : TPoint
2061: Function PointDist( const P1, P2 : TPoint) : Double;
2062: function PointDist(const P1,P2: TFloatPoint): Double;
2063: Function PointDist1( const P1, P2 : TFloatPoint) : Double;
2064: function PointDist2(const P1,P2: TPoint): Double;
2065: Function PointEqual( const P1, P2 : TPoint) : Boolean
2066: Function PointIsNull( const P : TPoint) : Boolean
2067: Function PointToLineSegmentDist( const Point, LineP1, LineP2 : TFloatPoint) : Double
2068: Function Poly( const X : Extended; const Coefficients : array of Double) : Extended
2069: Function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
2070: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
2071: Function Pop : Pointer
2072: Function Pop : TObject
2073: Function PopnStdDev( const Data : array of Double) : Extended
2074: Function PopnVariance( const Data : array of Double) : Extended
2075: Function PopulationVariance( const X : TDynFloatArray) : Float
2076: function Pos(SubStr, S: AnyString): Longint;
2077: Function PosEqual( const Rect : TRect) : Boolean
2078: Function PosEx( const SubStr, S : string; Offset : Integer) : Integer
2079: Function PosInSmallintArray( const ASearchInt : SmallInt; AArray : array of SmallInt) : Integer
2080: Function PosInStrArray(const SearchStr:string;Contents:array of string;const CaseSensitive:Boolean):Integer
2081: Function Post1( AURL : string; const ASource : TStrings) : string;
2082: Function Post2( AURL : string; const ASource : TStream) : string;
2083: Function Post3( AURL : string; const ASource : TIIdMultiPartFormDataStream) : string;
2084: Function PostData( const UserData : WideString; const CheckSum : DWORD) : Boolean
2085: Function PostData( const UserData : WideString; const CheckSum : integer): Boolean
2086: function PostMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2087: Function Power( const Base, Exponent : Extended) : Extended
2088: Function PowerBig(aval, n:integer): string;
2089: Function PowerIntJ( const X : Float; N : Integer) : Float;
2090: Function PowerJ( const Base, Exponent : Float) : Float;
2091: Function PowerOffOS : Boolean
2092: Function PreformatDateString( Ps : string) : string
2093: Function PresentValue(const Rate:Extend;NPeriods:Int;const Payment,
FutureVal:Extend;PaymentTime:TPaymentTime):Extended
2094: Function PrimeFactors( N : Cardinal) : TDynCardinalArray
2095: Function Printer : TPrinter

```

```

2096: Function ProcessPath2( const ABasePath:String; const APath: String; const APathDelim:string): string
2097: Function ProcessResponse : TIdHTTPWhatsNext
2098: Function ProduceContent : string
2099: Function ProduceContentFromStream( Stream : TStream) : string
2100: Function ProduceContentFromString( const S : string) : string
2101: Function ProgIDToClassID(const ProgID: string): TGUID;
2102: Function PromptDataLinkFile( ParentHandle : THandle; InitialFile : WideString) : WideString
2103: Function PromptDataSource( ParentHandle : THandle; InitialString : WideString) : WideString
2104: Function PromptForFileName( var AFileName : string; const AFiler : string; const ADefaultExt : string;
  const ATitle : string; const AInitialDir : string; SaveDialog: Boolean) : Boolean
2105: function PromptForFileName(var AFileName: string; const AFiler: string; const ADefaultExt: string;const
  ATitle: string; const AInitialDir: string; SaveDialog: Boolean): Boolean
2106: Function PSScriptNeedFile(Sender:TObject;const OriginFileName:String;var FileName,Output:String):Boolean
2107: Function PtInRect( const Rect : TRect; const P: TPoint) : Boolean
2108: function PtInRect(const Rect: TRect; const P: TPoint): Boolean
2109: Function Push( AItem : Pointer) : Pointer
2110: Function Push( AObject : TObject) : TObject
2111: Function Put1( AURL : string; const ASource : TStream) : string;
2112: Function Pythagoras( const X, Y : Extended) : Extended
2113: Function queryDLLInterface( var queryList : TStringList) : TStringList
2114: Function queryDLLInterfaceTwo( var queryList : TStringList) : TStringList
2115: Function QueryInterface(const IID: TGUID; out Obj): HResult, CdStdCall
2116: Function queryPerformanceCounter2(mse: int64): int64;
2117: //Function QueryPerformanceCounter(var lpPerformanceCount: Int64): LongBool; stdcall;
2118: //Function QueryPerformanceFrequency(mse: int64): boolean;
2119: Function QueryPerformanceCounter(var lcount: Int64): Boolean; stdcall;
2120: Function QueryPerformanceFrequency(var lfreq: int64): boolean; stdcall;
2121: Procedure QueryPerformanceCounter1(var aC: Int64);
2122: Function QueryPerformanceFrequency1(var freq: int64): boolean;
2123: Function Quote( const ACommand : String) : SmallInt
2124: Function QuotedStr( S : string) : string
2125: Function RadToCycle( const Radians : Extended) : Extended
2126: Function RadToDeg( const Radians : Extended) : Extended
2127: Function RadToDeg( const Value : Extended) : Extended;
2128: Function RadToDeg1( const Value : Double) : Double;
2129: Function RadToDeg2( const Value : Single) : Single;
2130: Function RadToGrad( const Radians : Extended) : Extended
2131: Function RadToGrad( const Value : Extended) : Extended;
2132: Function RadToGrad1( const Value : Double) : Double;
2133: Function RadToGrad2( const Value : Single) : Single;
2134: Function RandG( Mean, StdDev : Extended) : Extended
2135: function Random(const ARange: Integer): Integer;
2136: function random2(a: integer): double
2137: function RandomE: Extended;
2138: function RandomF: Extended;
2139: Function RandomFrom( const AValues : array of string) : string;
2140: Function RandomRange( const AFrom, ATo : Integer) : Integer
2141: function randSeed: longint
2142: Function RawToDataColumn( ACol : Integer) : Integer
2143: Function Read : Char
2144: Function Read( pv : Pointer; cb : Longint; pcbRead : PLongint) : HResult
2145: function Read(Buffer:String;Count:LongInt):LongInt
2146: Function ReadBinaryStream( const Section, Name : string; Value : TStream) : Integer
2147: Function ReadBool( const Section, Ident : string; Default : Boolean) : Boolean
2148: Function ReadCardinal( const AConvert : boolean) : Cardinal
2149: Function ReadChar : Char
2150: Function ReadClient( var Buffer, Count : Integer) : Integer
2151: Function ReadDate( const Section, Name : string; Default : TDateTime) : TDateTime
2152: Function ReadDateTime( const Section, Name : string; Default : TDateTime) : TDateTime
2153: Function ReadFloat( const Section, Name : string; Default : Double) : Double
2154: Function ReadFromStack(const ARaiseExceptfDisconnected:Bool;ATimeout:Int;const ARaiseExceptionTimeout:
  Bool):Int
2155: Function ReadInteger( const AConvert : boolean) : Integer
2156: Function ReadInteger( const Section, Ident : string; Default : Longint) : Longint
2157: Function ReadLn : string
2158: Function ReadLn( ATerminator : string; const ATimeout : Integer; AMaxLineLength : Integer) : string
2159: function Readln(question: string): string
2160: Function ReadLnWait( AFailCount : Integer) : string
2161: Function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
2162: Function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
2163: Function ReadSmallInt( const AConvert : boolean) : SmallInt
2164: Function ReadString( const ABytes : Integer) : string
2165: Function ReadString( const Section, Ident, Default : string) : string
2166: Function ReadString( Count : Integer) : string
2167: Function ReadTime( const Section, Name : string; Default : TDateTime) : TDateTime
2168: Function ReadTimeStampCounter : Int64
2169: Function RebootOS : Boolean
2170: Function Receive( ATimeOut : Integer) : TReplyStatus
2171: Function ReceiveBuf( var Buf, Count : Integer) : Integer
2172: Function ReceiveLength : Integer
2173: Function ReceiveText : string
2174: Function ReceiveSerialData(var Data: TByteArray; DataSize: cardinal): cardinal
2175: Function ReceiveSerialText: string
2176: Function RecodeDate( const AValue : TDateTime; const AYear, AMonth, ADay : Word) : TDateTime
2177: Function RecodeDateTime(const AValue:TDateTime;const AYear,AMonth,ADay,AHr,AMin,ASec,
  AMilliSec:Word):TDateTime
2178: Function RecodeDay( const AValue : TDateTime; const ADay : Word) : TDateTime
2179: Function RecodeHour( const AValue : TDateTime; const AHour : Word) : TDateTime
2180: Function RecodeMillisecond( const AValue : TDateTime; const AMilliSecond : Word) : TDateTime

```

```

2181: Function RecodeMinute( const AValue : TDateTime; const AMinute : Word ) : TDateTime
2182: Function RecodeMonth( const AValue : TDateTime; const AMonth : Word ) : TDateTime
2183: Function RecodeSecond( const AValue : TDateTime; const ASecond : Word ) : TDateTime
2184: Function RecodeTime( const AValue: TDateTime;const AHour,AMinute,ASecond,AMilliSecond:Word):TDateTime
2185: Function RecodeYear( const AValue : TDateTime; const AYear : Word ) : TDateTime
2186: Function Reconcile( const Results : OleVariant ) : Boolean
2187: Function Rect( Left, Top, Right, Bottom : Integer ) : TRect
2188: function Rect( ALeft: Integer; ATop: Integer; ARight: Integer; ABottom: Integer): TRect
2189: Function Rect2( const ATopLeft, ABottomRight : TPoint ) : TRect;
2190: Function RectAssign( const Left, Top, Right, Bottom : Integer ) : TRect
2191: Function RectAssignPoints( const TopLeft, BottomRight : TPoint ) : TRect
2192: Function RectBounds( const Left, Top, Width, Height : Integer ) : TRect
2193: Function RectCenter( const R : TRect ) : TPoint
2194: Function RectEqual( const R1, R2 : TRect ) : Boolean
2195: Function RectHeight( const R : TRect ) : Integer
2196: Function RectIncludesPoint( const R : TRect; const Pt : TPoint ) : Boolean
2197: Function RectIncludesRect( const R1, R2 : TRect ) : Boolean
2198: Function RectIntersection( const R1, R2 : TRect ) : TRect
2199: Function RectIntersectRect( const R1, R2 : TRect ) : Boolean
2200: Function RectIsEmpty( const R : TRect ) : Boolean
2201: Function RectIsNull( const R : TRect ) : Boolean
2202: Function RectIsSquare( const R : TRect ) : Boolean
2203: Function RectIsValid( const R : TRect ) : Boolean
2204: Function RectsAreValid( R : array of TRect ) : Boolean
2205: Function RectUnion( const R1, R2 : TRect ) : TRect
2206: Function RectWidth( const R : TRect ) : Integer
2207: Function RedComponent( const Color32 : TColor32 ) : Integer
2208: Function Refresh : Boolean
2209: Function RefStringListCopy(aRefArray:TStringlist):TStringList;
2210: Function RegisterConversionFamily( const ADescription : string ) : TConvFamily
2211: Function RegisterConversionType( AConvTypeInfo : TConvTypeInfo; out AType : TConvType ) : Boolean;
2212: Function RegisterConversionType(const AFam:TConvFamil;const ADescr:string;const AFact:Double):TConvType
2213: Function RegistryRead(keyHandle: Longint; keyPath, myField: String): string;
2214: Function ReleaseDC(hdwd: HWND; hdc: HDC): integer;
2215: Function ReleaseHandle : HBITMAP
2216: Function ReleaseHandle : HENHMETAFILE
2217: Function ReleaseHandle : HICON
2218: Function ReleasePalette : HPALETTE
2219: Function RemainderFloat( const X, Y : Float ) : Float
2220: Function Remove( AClass : TClass ) : Integer
2221: Function Remove( AComponent : TComponent ) : Integer
2222: Function Remove( AItem : Integer ) : Integer
2223: Function Remove( AItem : Pointer ) : Pointer
2224: Function Remove( AItem : TObject ) : TObject
2225: Function Remove( AObject : TObject ) : Integer
2226: Function RemoveBackslash( const PathName : string ) : string
2227: Function RemoveDF( aString : String ) : String //removes thousand separator
2228: Function RemoveDir( Dir : string ) : Boolean
2229: function RemoveDir(const Dir: string): Boolean
2230: Function RemoveDirectory(PathName: PChar): WordBool; stdcall;
2231: Function RemoveFileExt( const FileName : string ) : string
2232: Function RemoveHeaderEntry( AHeader, AEntry : string ) : string
2233: Function RenameFile( OldName, NewName : string ) : Boolean
2234: function RenameFile(const OldName: string; const NewName: string): Boolean
2235: Function ReplaceStr( const AText, AFromText, AToText : string ) : string
2236: Function ReplaceText( const AText, AFromText, AToText : string ) : string
2237: Function Replicate(c : char;I : longInt) : String;
2238: Function Request : TWebRequest
2239: Function ResemblesText( const AText, AOther : string ) : Boolean
2240: Function Reset : Boolean
2241: function Reset2(mypath: string):string;
2242: Function ResInstLoad(Instance:THandle;ResType:TResType; const Name:string;MaskColor: TColor): Boolean
2243: Function ResourceLoad( ResType : TResType; const Name : string; MaskColor : TColor ) : Boolean
2244: Function Response : TWebResponse
2245: Function ResumeSupported : Boolean
2246: Function REthinkHotkeys : BOOLEAN
2247: Function REthinkLines : BOOLEAN
2248: Function Retrieve( const MsgNum : Integer; AMsg : TIIdMessage ) : Boolean
2249: Function RetrieveCurrentDir : string
2250: Function RetrieveDeltas( const cdsArray : array of TClientDataset ) : Variant
2251: Function RetrieveHeader( const MsgNum : Integer; AMsg : TIIdMessage ) : Boolean
2252: Function RetrieveMailBoxSize : integer
2253: Function RetrieveMsgSize( const MsgNum : Integer ) : Integer
2254: Function RetrieveProviders( const cdsArray : array of TClientDataset ) : Variant
2255: Function RetrieveRaw( const MsgNum : Integer; const Dest : TStrings ) : boolean
2256: Function ReturnMIMETYPE( var MediaType, EncType : String ) : Boolean
2257: Function ReverseBits( Value : Byte ) : Byte;
2258: Function ReverseBits1( Value : Shortint ) : Shortint;
2259: Function ReverseBits2( Value : Smallint ) : Smallint;
2260: Function ReverseBits3( Value : Word ) : Word;
2261: Function ReverseBits4( Value : Cardinal ) : Cardinal;
2262: Function ReverseBits4( Value : Integer ) : Integer;
2263: Function ReverseBits5( Value : Int64 ) : Int64;
2264: Function ReverseBytes( Value : Word ) : Word;
2265: Function ReverseBytes1( Value : Smallint ) : Smallint;
2266: Function ReverseBytes2( Value : Integer ) : Integer;
2267: Function ReverseBytes3( Value : Cardinal ) : Cardinal;
2268: Function ReverseBytes4( Value : Int64 ) : Int64;
2269: Function ReverseString( const AText : string ) : string

```

```

2270: Function ReverseDNSLookup( const IPAddrs:String; DNSServer:String; Timeout,Retries:Int; var
2271:   HostName:String ):Bool;
2272: Function Revert : HRESULT;
2273: Function RGB(R,G,B: Byte): TColor;
2274: Function RGB2BGR( const Color : TColor ) : TColor;
2275: Function RGB2TColor( R, G, B : Byte ) : TColor;
2276: Function RGBToWebColorName( RGB : Integer ) : string;
2277: Function RgbToHtml( Value : TColor ) : string;
2278: Function HtmlToRgb(const Value: string): TColor;
2279: Function RightStr( const AStr : String; Len : Integer ) : String;
2280: Function RightStr( const AText : AnsiString; const ACount : Integer ) : AnsiString;
2281: Function RightStr( const AText : WideString; const ACount : Integer ) : WideString;
2282: Function ROL( Aval : LongWord; AShift : Byte ) : LongWord;
2283: Function ROR( Aval : LongWord; AShift : Byte ) : LongWord;
2284: Function RotatePoint( Point : TFloatPoint; const Center : TFloatPoint; const Angle : Float ) : TFloatPoint;
2285: function RotatePoint(Point: TFloatPoint; const Center: TFloatPoint; const Angle: Double): TFloatPoint;
2286: Function Round(e : Extended) : Longint;
2287: Function Round64(e: extended): Int64;
2288: Function RoundAt( const Value : string; Position : SmallInt ) : string;
2289: type TRoundToRange = -37..37; TRoundToEXRangeExtended = -20..20;
2290: Function RoundTo( const AValue: Extended; const ADigit: TRoundToEXRangeExtended): Extended; ';
2291: Function SimpleRoundTo( const AValue: Extended; const ADigit: TRoundToRange): Extended; ';
2292: Function RoundFrequency( const Frequency : Integer ) : Integer;
2293: Function RoundInt( Value : Integer; StepSize : Integer ) : Integer;
2294: Function RoundPoint( const X, Y : Double ) : TPoint;
2295: Function RoundRect( const ALeft, ATop, ARight, ABottom : Double ) : TRect;
2296: Function RowCount : Integer;
2297: Function RowRequest( const Row : OleVariant; RequestType : Integer; var OwnerData : OleVariant): OleVariant;
2298: Function RowRequest( Row : OleVariant; Options : TFetchOptions ) : OleVariant;
2299: Function RPos( const ASub, AIn : String; AStart : Integer ) : Integer;
2300: Function RRot( const Value : Byte; const Count : TBitRange ) : Byte;
2301: Function RRot1( const Value : Word; const Count : TBitRange ) : Word;
2302: Function RRot2( const Value : Integer; const Count : TBitRange ) : Integer;
2303: Function RunDLL32( const ModuleNa, FuncName, Cmdline:string; WaitForCompletion:Bool; CmdShow:Integer ):Boolean;
2304: Function RunningProcessesList( const List : TStrings; FullPath : Boolean ) : Boolean;
2305: Function S_AddBackSlash( const ADirName : string ) : string;
2306: Function S_AllTrim( const cStr : string ) : string;
2307: Function S_AtRepl( const cAT, cStr, cRepl : string ) : string;
2308: Function S_Cut( const cStr : string; const iLen : integer ) : string;
2309: Function S_DecryptCRC32( const crc : string; StartKey, MultKey, AddKey : integer ) : integer;
2310: Function S_DirExists( const ADir : string ) : Boolean;
2311: Function S_Empty( const cStr : string ) : boolean;
2312: Function S_EncryptCRC32( const crc : LongWORD; StartKey, MultKey, AddKey : integer ) : string;
2313: Function S_LargeFontsActive : Boolean;
2314: Function S_LimitDigits( AValue : Extended; ANumDigits : Integer ) : Extended;
2315: Function S_LTrim( const cStr : string ) : string;
2316: Function S_ReadNextTextLineFromStream( stream : TStream ) : string;
2317: Function S_RepeatChar( const iLen : integer; const AChar : Char ) : String;
2318: Function S_ReplFirst( const cAT, cStr, cRepl : string ) : string;
2319: Function S_RoundDecimal( AValue : Extended; APPlaces : Integer ) : Extended;
2320: Function S_RTrim( const cStr : string ) : string;
2321: Function S_RTrimCopy( const cStr : string; iPos, iLen : integer ) : string;
2322: //Type TS_ShellExecuteCmd = (seCmdOpen, seCmdPrint, seCmdExplore);
2323: Function S_ShellExecute( aFilename : string; aParameters : string; aCommand : TS_ShellExecuteCmd ) : string;
2324: Function S_Space( const iLen : integer ) : String;
2325: Function S_StrBlanks( const cStr : string; const iLen : integer ) : string;
2326: Function S_StrBlanksCuttooLong( const cStr : string; const iLen : integer ) : string;
2327: Function S_StrCRC32( const Text : string ) : LongWORD;
2328: Function S_StrDecrypt96( const InString : string; StartKey, MultKey, AddKey : Integer ) : string;
2329: Function S_StrEncrypt96( const InString : string; StartKey, MultKey, AddKey : Integer ) : string;
2330: Function S_StringToUTF_8( const AString : string ) : string;
2331: Function S_StrLBlanks( const cStr : string; const iLen : integer ) : string;
2332: function S_StrToReal( const cStr: string; var R: Double): Boolean;
2333: Function S_TokenEnd( cBuffer : PChar; lEmptyToken : boolean ) : boolean;
2334: Function S_TokenNext( cBuffer : PChar; lEmptyToken : boolean ) : string;
2335: Function S_UTF_8ToString( const AString : string ) : string;
2336: Function S_WBox( const AText : string ) : integer;
2337: Function SameDate( const A, B : TDateTime ) : Boolean;
2338: function SameDate( const A, B: TDateTime): Boolean;
2339: Function SameDateTime( const A, B : TDateTime ) : Boolean;
2340: function SameDateTime( const A, B: TDateTime): Boolean;
2341: Function SameFileName( S1, S2 : string ) : Boolean;
2342: Function SameText( S1, S2 : string ) : Boolean;
2343: function SameText( const S1: string; const S2: string): Boolean;
2344: Function SameTime( const A, B : TDateTime ) : Boolean;
2345: function SameTime( const A, B: TDateTime): Boolean;
2346: function SameValue( const A, B: Extended; Epsilon: Extended): Boolean //overload;
2347: function SameValue1( const A, B: Double; Epsilon: Double): Boolean //overload;
2348: function SameValue2( const A, B: Single; Epsilon: Single): Boolean //overload;
2349: Function SampleVariance( const X : TDynFloatArray ) : Float;
2350: Function Sar( const Value : Shortint; const Count : TBitRange ) : Shortint;
2351: Function Sar1( const Value : Smallint; const Count : TBitRange ) : Smallint;
2352: Function Sar2( const Value : Integer; const Count : TBitRange ) : Integer;
2353: Function SaveAsExcelFile( const AfileName : TFileName ) : Boolean;
2354: Function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
2355: Function SaveAsExcel(aGrid: TStringGrid; aSheetName, afileName: string; openexcel: boolean): Boolean;
2356: Function ScanF( const aformat: String; const args: array of const): string;
2357: Function SCREENTOCLIENT(POINT:TPOINT):TPOINT

```

```

2358: Function SearchBuf(Buf:PChar; BufLen:Integer; SelStart,SelLength:Integer;SearchString:String;Options: TStringSearchOptions):PChar
2359: Function SearchBuf2(Buf: String;SelStart,SelLength:Integer; SearchString: String;Options:TStringSearchOptions):Integer;
2360: function SearchRecattr: integer;
2361: function SearchRecExcludeAttr: integer;
2362: Function SearchRecFileSize64( const SearchRec : TSearchRec ) : Int64
2363: function SearchRecname: string;
2364: function SearchRecsize: integer;
2365: function SearchRecTime: integer;
2366: Function Sec( const X : Extended ) : Extended
2367: Function Secant( const X : Extended ) : Extended
2368: Function SecH( const X : Extended ) : Extended
2369: Function SecondOf( const AValue : TDateTime ) : Word
2370: Function SecondOfTheDay( const AValue : TDateTime ) : LongWord
2371: Function SecondOfTheHour( const AValue : TDateTime ) : Word
2372: Function SecondOfTheMinute( const AValue : TDateTime ) : Word
2373: Function SecondOfTheMonth( const AValue : TDateTime ) : LongWord
2374: Function SecondOfTheWeek( const AValue : TDateTime ) : LongWord
2375: Function SecondOfTheYear( const AValue : TDateTime ) : LongWord
2376: Function SecondsBetween( const ANow, AThen : TDateTime ) : Int64
2377: Function SecondSpan( const ANow, AThen : TDateTime ) : Double
2378: Function SectionExists( const Section : string ) : Boolean
2379: Function Seek( const KeyValues: Variant; SeekOption : TSeekOption ) : Boolean
2380: Function Seek( dlibMove : Longint; dwOrigin : Longint; out libNewPosition : Largeint ) : HResult
2381: function Seek(Offset:LongInt;Origin:Word):LongInt
2382: Function SelectDirectory( var Directory : string; Options : TSelectDirOpts; HelpCtx:Longint): Boolean;
2383: Function SelectDirectory( const Caption : string; const Root : WideString; var Directory : string; Options : TSelectDirExtOpts; Parent : TWinControl ) : Boolean;
2384: Function SelectImage( var AFileName : string; const Extensions, Filter : string ) : Boolean
2385: function SendAppMessage(Msg: Cardinal; WParam, LParam: Longint): Longint
2386: Function SendBuf( var Buf, Count : Integer ) : Integer
2387: Function SendCmd( const AOut : string; const AResponse : SmallInt ) : SmallInt;
2388: Function SendCmd( const AOut : string; const AResponse : array of SmallInt ) : SmallInt;
2389: Function SendKey( AppName : string; Key : Char ) : Boolean
2390: function SendMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2391: Function SendStream( AStream : TStream ) : Boolean
2392: Function SendStreamThenDrop( AStream : TStream ) : Boolean
2393: Function SendText( const S : string ) : Integer
2394: Function SendSerialData(Data: TByteArray; DataSize: cardinal): cardinal
2395: Function SendSerialText(Data: String): cardinal
2396: Function Sent : Boolean
2397: Function ServicesFilePath: string
2398: Function SetAlpha( const Color32 : TColor32; NewAlpha : Integer ) : TColor32
2399: Function SetBit( const Value : Byte; const Bit : TBitRange ) : Byte;
2400: Function SetBit1( const Value : Shortint; const Bit : TBitRange ) : Shortint;
2401: Function SetBit2( const Value : Smallint; const Bit : TBitRange ) : Smallint;
2402: Function SetBit3( const Value : Word; const Bit : TBitRange ) : Word;
2403: Function SetBit4( const Value : Cardinal; const Bit : TBitRange ) : Cardinal;
2404: Function SetBit4( const Value : Integer; const Bit : TBitRange ) : Integer;
2405: Function SetBit5( const Value : Int64; const Bit : TBitRange ) : Int64;
2406: Function SetClipboard( NewClipboard : TClipboard ) : TClipboard
2407: Function SetColorBlue( const Color : TColor; const Blue : Byte ) : TColor
2408: Function SetColorFlag( const Color : TColor; const Flag : Byte ) : TColor
2409: Function SetColorGreen( const Color : TColor; const Green : Byte ) : TColor
2410: Function SetColorRed( const Color : TColor; const Red : Byte ) : TColor
2411: Function SetCurrentDir( Dir : string ) : Boolean
2412: function SetCurrentDir(const Dir: string): Boolean
2413: Function SetCurrentDirectory(PathName: PChar): WordBool; stdcall;
2414: Function SetDirCreation( const DirName : string; const DateTime : TDateTime ) : Boolean
2415: Function SetDirLastAccess( const DirName : string; const DateTime : TDateTime ) : Boolean
2416: Function SetDirLastWrite( const DirName : string; const DateTime : TDateTime ) : Boolean
2417: Function SetDisplayResolution( const XRes, YRes : DWORD ) : Longint
2418: Function SetEndOfFile(Handle: Integer): LongBool; stdcall;
2419: Function SetEnvironmentVar( const Name, Value : string ) : Boolean
2420: Function SetErrorProc( ErrorProc : TSocketErrorProc ) : TSocketErrorProc
2421: Function SetFileCreation( const FileName : string; const DateTime : TDateTime ) : Boolean
2422: Function SetFileLastAccess( const FileName : string; const DateTime : TDateTime ) : Boolean
2423: Function SetFileLastWrite( const FileName : string; const DateTime : TDateTime ) : Boolean
2424: Function SetFileTimeStamp( const FileName : string; TimeStamp : Integer ) : Boolean
2425: function SETFOCUSDECONTROL(CONTROL:TWINCONTROL):BOOLEAN
2426: Function SetLocalTime( Value : TDateTime ) : boolean
2427: Function SetPrecisionTolerance( NewTolerance : Float ) : Float
2428: Function SetPrinter( NewPrinter : TPrinter ) : TPrinter
2429: Function SetPrivilege(privilegeName: string; enable: boolean): boolean;
2430: Function SetRGBValue( const Red, Green, Blue : Byte ) : TColor
2431: Function SetSequence( S, Localizar, Substituir : shortstring ) : shortstring
2432: Function SetSize( libNewSize : Longint ) : HResult
2433: Function SetUserObjectFullAccess( hUserObject : THandle ) : Boolean
2434: Function Sgn( const X : Extended ) : Integer
2435: function SHA1(const fileName: string): string;
2436: function SHA256(astr: string; amode: char): string)
2437: function SHA512(astr: string; amode: char): string)
2438: Function ShareMemoryManager : Boolean
2439: function ShellExecute(hWnd:HWND;Operation,FileN,Parameters,Dir:string;ShowCmd:Integer):integer;stdcall;
2440: function ShellExecute2(hwnd: HWND; const FileName: string):integer; stdcall;
2441: Function ShellExecute3(afilename: string; aParameters: string; aCommand:TS_ShellExecuteCmd): string;
2442: Function SHORTCUT( KEY : WORD; SHIFT : TSHIFTSTATE ) : TSHORTCUT
2443: Function SHORTCUTTOTEXT( SHORTCUT : TSHORTCUT ) : String

```

```

2444: function ShortDateFormat: string;
2445: Function ShortenString(const DC:HDC;const S:WideString;const Width:Int;const
  RTL:Boolean;EllipsisWidth:Int):WideString
2446: function ShortTimeFormat: string;
2447: function SHOWMODAL: INTEGER
2448: Function ShowModalControl(aControl : TControl; BS : TFormBorderStyle; BI : TBorderIcons; WS :
  TWindowState; aColor : TColor; BW : Integer; Title : String; BeforeShowModal : TNotifyEvent) : TModalResult';
2449: Function ShowModalPanel(aPnl:TCustomPanel;Title:String;ShowCloseIcn:Bool;BefShowModal:TNotifyEvent):TModalResult;
2450: function ShowWindow(C1: HWND; C2: integer): boolean;
2451: procedure ShowMemory //in Dialog
2452: function ShowMemory2: string;
2453: Function ShutDownOS : Boolean
2454: Function Signe( const X, Y : Extended ) : Extended
2455: Function Sign( const X : Extended ) : Integer
2456: Function Sin(e : Extended) : Extended;
2457: Function sinc( const x : Double ) : Double
2458: Function SinJ( X : Float ) : Float
2459: Function Size( const AFileName : String ) : Integer
2460: function Sizeof: Longint;
2461: Function SizeofResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle ) : Integer
2462: function SlashSep(const Path, S: String): String
2463: Function SLNDepreciation( const Cost, Salvage : Extended; Life : Integer ) : Extended
2464: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL ) : DWORD
2465: Function SmallPoint(X, Y: Integer): TSmallPoint)
2466: Function Soundex( const AText : string; ALength : TSoundexLength ) : string
2467: Function SoundexCompare( const AText, AOther : string; ALength : TSoundexLength ) : Integer
2468: Function SoundexInt( const AText : string; ALength : TSoundexIntLength ) : Integer
2469: Function SoundexProc( const AText, AOther : string ) : Boolean
2470: Function SoundexSimilar( const AText, AOther : string; ALength : TSoundexLength ) : Boolean
2471: Function SoundexWord( const AText : string ) : Word
2472: Function SourcePos : Longint
2473: function SourcePos:LongInt
2474: Function Split0( Str : string; const substr : string ) : TStringList
2475: Procedure SplitNameValue( const Line : string; var Name, Value : string )
2476: Function SQLRequiresParams( const SQL : WideString ) : Boolean
2477: Function Sqr(e : Extended) : Extended;
2478: Function Sqrt(e : Extended) : Extended;
2479: Function StartIP : String
2480: Function StartPan( WndHandle : THandle; AControl : TControl ) : Boolean
2481: Function StartOfADay( const AYear, AMonth, ADay : Word ) : TDateTime;
2482: Function StartOfADay1( const AYear, ADayOfYear : Word ) : TDateTime;
2483: Function StartOfAMonth( const AYear, AMonth : Word ) : TDateTime
2484: Function StartOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word ) : TDateTime
2485: Function StartOfAYear( const AYear : Word ) : TDateTime
2486: Function StartOfTheDay( const AValue : TDateTime ) : TDateTime
2487: Function StartOfTheMonth( const AValue : TDateTime ) : TDateTime
2488: Function StartOfTheWeek( const AValue : TDateTime ) : TDateTime
2489: Function StartOfTheYear( const AValue : TDateTime ) : TDateTime
2490: Function StartsStr( const ASubText, AText : string ) : Boolean
2491: Function StartsText( const ASubText, AText : string ) : Boolean
2492: Function StartsWith( const ANSIStr, APattern : String ) : Boolean
2493: Function StartsWith( const str : string; const sub : string ) : Boolean
2494: Function StartsWithACE( const ABytes : TIdBytes ) : Boolean
2495: Function StatusString( StatusCode : Integer ) : string
2496: Function StdDev( const Data : array of Double ) : Extended
2497: Function Stop : Float
2498: Function StopCount( var Counter : TJclCounter ) : Float
2499: Function StoreColumns : Boolean
2500: Function StrAfter( const sString : string; const sDelimiters : string ) : string;
2501: Function StrAfter1( const sString : string; const sDelimiters : string; out cDelimiter : char ) : string;
2502: Function StrAlloc( Size : Cardinal ) : PChar
2503: function StrAlloc(Size: Cardinal): PChar)
2504: Function StrBefore( const sString : string; const sDelimiters : string ) : string;
2505: Function StrBefore1( const sString : string; const sDelimiters:string; out cDelimiter:char): string;
2506: Function StrBufSize( Str : PChar ) : Cardinal
2507: function StrBufSize(const Str: PChar): Cardinal)
2508: Function StrByteType( Str : PChar; Index : Cardinal ) : TMbcsByteType
2509: function StrByteType(Str: PChar; Index: Cardinal): TMbcsByteType)
2510: Function StrCat( Dest : PChar; Source : PChar ) : PChar
2511: function StrCat(Dest: PChar; const Source: PChar): PChar)
2512: Function StrCharLength( Str : PChar ) : Integer
2513: Function StrComp( Str1, Str2 : PChar ) : Integer
2514: function StrComp(const Str1: PChar; const Str2: PChar): Integer)
2515: Function StrCopy( Dest : PChar; Source : PChar ) : PChar
2516: function StrCopy(Dest: PChar; const Source: PChar): PChar)
2517: Function Stream_to_AnsiString( Source : TStream ) : ansistring
2518: Function Stream_to_Base64( Source : TStream ) : ansistring
2519: Function Stream_to_decimalbytes( Source : TStream ) : string
2520: Function Stream2WideString( oStream : TStream ) : WideString
2521: Function StreamtoAnsiString( Source : TStream ) : ansistring
2522: Function StreamToByte( Source : TStream ) : string
2523: Function Stream.ToDecimalbytes( Source : TStream ) : string
2524: Function StreamtoOrd( Source : TStream ) : string
2525: Function StreamToString( Source : TStream ) : string
2526: Function StrECopy( Dest : PChar; Source : PChar ) : PChar
2527: Function StrEmpty( const sString : string ) : boolean
2528: Function StrEnd( Str : PChar ) : PChar

```

```

2529: function StrEnd(const Str: PChar): PChar
2530: Function StrFilter( const sString : string; xValidChars : TCharSet ) : string
2531: Function StrFmt(Buffer, Format: PChar; const Args: array of const): PChar
2532: Function StrGet(var S : String; I : Integer) : Char;
2533: Function StrGet2(S : String; I : Integer) : Char;
2534: Function StrHasPrefix( const sString : string; const sPrefix : string) : boolean
2535: Function StrHasSuffix( const sString : string; const sSuffix : string) : boolean
2536: Function StrHtmlDecode( const AStr : String) : String
2537: Function StrHtmlEncode( const AStr : String) : String
2538: Function StrToBytes(const Value: String): TBytes;
2539: Function StrIComp( Str1, Str2 : PChar) : Integer
2540: Function StringOfChar(c : char;I : longInt) : String;
2541: Function StringOfChar2( ch : WideChar; Count : Integer) : WideString;
2542: Function StringPad(InputStr,FillChar: String; StrLen:Integer; StrJustify:Boolean): String;
2543: Function StringRefCount(const s: String): integer;
2544: Function StringReplace( S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2545: Function JStringReplace( const S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2546: Function StringReplace(const SourceString, OldPattern,NewPattern:string; Flags:TReplaceFlags): string;
2547: Function StringRemove( const S, Pattern : string; Flags : TReplaceFlags) : string
2548: Function StringToBoolean( const Ps : string) : Boolean
2549: function StringToColor(const S: string): TColor
2550: function StringToCursor(const S: string): TCursor;
2551: function StringToGUID(const S: string): TGUID
2552: Function StringTokenizer( const str : string; const delim : string) : IStringTokenizer
2553: Function StringToStringArray( const str : string; const delim : string) : TStringDynArray
2554: Function StringWidth( S : string) : Integer
2555: Function StrInternetToDateTIme( Value : string) : TDateTIme
2556: Function StrIsDateTIme( const Ps : string) : Boolean
2557: Function StrIsFloatMoney( const Ps : string) : Boolean
2558: Function StrIsInteger( const S : string) : Boolean
2559: Function StrLCat( Dest : PChar; Source : PChar; MaxLen : Cardinal) : PChar
2560: Function StrLComp( Str1, Str2 : PChar; MaxLen : Cardinal) : Integer
2561: Function StrLCopy( Dest : PChar; Source : PChar; MaxLen : Cardinal) : PChar
2562: Function StrLen( Str : PChar) : Cardinal
2563: function StrLen(const Str: PChar): Cardinal
2564: Function StrLessPrefix( const sString : string; const sPrefix : string) : string
2565: Function StrLessSufix( const sString : string; const sSuffix : string) : string
2566: Function StrLIComp( Str1, Str2 : PChar; MaxLen : Cardinal) : Integer
2567: Function StrLower( Str : PChar) : PChar
2568: Function StrMove( Dest : PChar; Source : PChar; Count : Cardinal) : PChar
2569: function StrMove(Dest: PChar; const Source: PChar; Count: Cardinal): PChar)
2570: Function StrNew( Str : PChar) : PChar
2571: function StrNew(const Str: PChar): PChar)
2572: Function StrNextChar( Str : PChar) : PChar
2573: Function StrPad( const sString : string; const sPad : string; const iLength : integer) : string
2574: Function StrParse( var sString : string; const sDelimiters : string) : string;
2575: Function StrParseL( var sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2576: Function StrPas( Str : PChar) : string
2577: function StrPas(const Str: PChar): string
2578: Function StrPCopy( Dest : PChar; Source : string) : PChar
2579: function StrPLCopy( Dest : PChar; Source : string): PChar)
2580: Function StrPos( Str1, Str2 : PChar) : PChar
2582: Function StrScan(const Str: PChar; Chr: Char): PChar)
2583: Function StrRScan(const Str: PChar; Chr: Char): PChar)
2584: Function StrToBcd( const AValue : string) : TBcd
2585: Function StrToBool( S : string) : Boolean
2586: Function StrToBoolDef( S : string; Default : Boolean) : Boolean
2587: Function StrToCard( const AStr : String) : Cardinal
2588: Function StrToConv( AText : string; out AType : TConvType) : Double
2589: Function StrToCurr( S : string) : Currency;
2590: function StrToCurr(const S: string): Currency)
2591: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2592: Function StrToDate( S : string) : TDateTIme;
2593: function StrToDate(const s: string): TDateTIme;
2594: Function StrToDateDef( S : string; Default : TDateTIme) : TDateTIme;
2595: Function StrToDateTIme( S : string) : TDateTIme;
2596: function StrToDateTIme(const S: string): TDateTIme)
2597: Function StrToDateTImeDef( S : string; Default : TDateTIme) : TDateTIme;
2598: Function StrToDay( const ADay : string) : Byte
2599: Function StrToFloat( S : string) : Extended;
2600: function StrToFloat(s: String): Extended;
2601: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2602: function StrToFloatDef(const S: string; const Default: Extended): Extended)
2603: Function StrToFloat( S : string) : Extended;
2604: Function StrToFloat2( S : string; FormatSettings : TFormatSettings) : Extended;
2605: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2606: Function StrToFloatDef2(S: string; Default: Extended;FormatSettings:TFormatSettings): Extended;
2607: Function StrToCurr( S : string) : Currency;
2608: Function StrToCurr2( S : string; FormatSettings : TFormatSettings) : Currency;
2609: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2610: Function StrToCurrDef2( S : string; Default : Currency; FormatSettings : TFormatSettings) : Currency;
2611: Function StrToTime2( S : string; FormatSettings : TFormatSettings) : TDateTIme;
2612: Function StrToTimeDef( S : string; Default : TDateTIme) : TDateTIme;
2613: Function StrToTimeDef2( S : string; Default : TDateTIme; FormatSettings:TFormatSettings):TDateTIme;
2614: Function TryStrToTime( S : string; Value : TDateTIme) : Boolean;
2615: Function StrToDateTIme( S : string) : TDateTIme;
2616: Function StrToDateTIme2( S : string; FormatSettings : TFormatSettings) : TDateTIme;
2617: Function StrToDateTImeDef( S : string; Default : TDateTIme) : TDateTIme;

```

```

2618: Function StrToFloatRegionalIndependent(aValue: String;aDecimalSymbol:Char;aDigitGroupSymbol:Char): Extended
2619: Function StrToInt( S : string ) : Integer
2620: function StrToInt(s: String): Longint;
2621: Function StrToInt64( S : string ) : Int64
2622: function StrToInt64(s: String): int64;
2623: Function StrToInt64Def( S : string; Default : Int64 ) : Int64
2624: function StrToInt64Def(const S: string; const Default: Int64):Int64)
2625: Function StrToIntDef( S : string; Default : Integer ) : Integer
2626: function StrToIntDef(const S: string; Default: Integer): Integer)
2627: function StrToIntDef(s: String; def: Longint): Longint;
2628: Function StrToMonth( const AMonth : string ) : Byte
2629: Function StrToTime( S : string ) : TDateTime;
2630: function StrToTime(const S: string): TDateTime)
2631: Function StrToTimeDef( S : string; Default : TDateTime ) : TDateTime;
2632: Function StrToWord( const Value : String ) : Word
2633: Function StrToXmlDate( const DateStr : string; const Format : string ) : string
2634: Function StrToXmlDateTime( const DateStr : string; const Format : string ) : string
2635: Function StrToXmlTime( const TimeStr : string; const Format : string ) : string
2636: Function StrUpper( Str : PChar ) : PChar
2637: Function StuffString( const AText : string; AStart, ALen : Cardinal; const ASubText : string ) : string
2638: Function Sum( const Data : array of Double ) : Extended
2639: Function SumFloatArray( const B : TDynFloatArray ) : Float
2640: Function SumInt( const Data : array of Integer ) : Integer
2641: Function SumOfSquares( const Data : array of Double ) : Extended
2642: Function SumPairProductFloatArray( const X, Y : TDynFloatArray ) : Float
2643: Function SumSquareDiffFloatArray( const B : TDynFloatArray; Diff : Float ) : Float
2644: Function SumSquareFloatArray( const B : TDynFloatArray ) : Float
2645: Function Supports( CursorOptions : TCursorOptions ) : Boolean
2646: Function SupportsClipboardFormat( AFormat : Word ) : Boolean
2647: Function SwapWord(w : word): word)
2648: Function SwapInt(i : integer): integer)
2649: Function SwapLong(L : longint): longint)
2650: Function Swap(i : integer): integer)
2651: Function SYDDepreciation( const Cost, Salvage : Extended; Life, Period : Integer ) : Extended
2652: Function SyncTime : Boolean
2653: Function SysErrorMessage( ErrorCode : Integer ) : string
2654: function SysErrorMessage(ErrorCode: Integer): string)
2655: Function SystemTimeToDate( SystemTime : TSystemTime ) : TDateTime
2656: function SystemTimeToDate( const SystemTime: TSystemTime): TDateTime;
2657: Function SysStringLen(const S: WideString): Integer; stdcall;
2658: Function TabRect( Index : Integer ) : TRect
2659: Function Tan( const X : Extended ) : Extended
2660: Function TaskMessageDlg(const Title,
  Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
2661: Function TaskMessageDlgl( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
  HelpCtx : Longint; DefaultButton : TMsgDlgBtn) : Integer;
2662: Function TaskMessageDlgPos( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
  HelpCtx : Longint; X, Y : Integer ) : Integer;
2663: Function TaskMessageDlgPos1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
  HelpCtx : Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn ) : Integer;
2664: Function TaskMessageDlgPosHelp( const Title, Msg : string; DlgType : TMsgDlgType; Buttons :
  TMsgDlgButtons; HelpCtx : Longint; X, Y : Integer; const HelpFileName : string ) : Integer;
2665: Function TaskMessageDlgPosHelp1( const Title, Msg:string;DlgType: TMsgDlgType; Buttons : TMsgDlgButtons;
  HelpCtx:Longint; X,Y: Integer;const HelpFileName:string;DefaultButton:TMsgDlgBtn): Integer;
2666: Function TenToY( const Y : Float ) : Float
2667: Function TerminateApp( ProcessID : DWORD; Timeout : Integer ) : TJclTerminateAppResult
2668: Function TerminateTask( Wnd : HWND; Timeout : Integer ) : TJclTerminateAppResult
2669: Function TestBit( const Value : Byte; const Bit : TBitRange ) : Boolean;
2670: Function TestBit2( const Value : Shortint; const Bit : TBitRange ) : Boolean;
2671: Function TestBit3( const Value : Smallint; const Bit : TBitRange ) : Boolean;
2672: Function TestBit4( const Value : Word; const Bit : TBitRange ) : Boolean;
2673: Function TestBit5( const Value : Cardinal; const Bit : TBitRange ) : Boolean;
2674: Function TestBit6( const Value : Integer; const Bit : TBitRange ) : Boolean;
2675: Function TestBit7( const Value : Int64; const Bit : TBitRange ) : Boolean;
2676: Function TestBits( const Value, Mask : Byte ) : Boolean;
2677: Function TestBits1( const Value, Mask : Shortint ) : Boolean;
2678: Function TestBits2( const Value, Mask : Smallint ) : Boolean;
2679: Function TestBits3( const Value, Mask : Word ) : Boolean;
2680: Function TestBits4( const Value, Mask : Cardinal ) : Boolean;
2681: Function TestBits5( const Value, Mask : Integer ) : Boolean;
2682: Function TestBits6( const Value, Mask : Int64 ) : Boolean;
2683: Function TestFDIVInstruction : Boolean
2684: function TestStreamFormat(Stream: TStream): TStreamOriginalFormat
2685: Function TextExtent( const Text : string ) : TSize
2686: function TextHeight(Text: string): Integer;
2687: Function TextIsSame( const A1 : string; const A2 : string ) : Boolean
2688: Function TextStartsWith( const S, SubS : string ) : Boolean
2689: function TextToFloat(Buffer: PChar; var Value: Extended; ValueType: TFloatingValue): Boolean)
2690: Function ConvInteger(i : integer):string;
2691: Function IntegerToText(i : integer):string;
2692: Function TEXTTOSHORTCUT( TEXT : String ) : TSHORTCUT
2693: function TextWidth(Text: string): Integer;
2694: Function ThreadCount : integer
2695: function ThousandSeparator: char;
2696: Function Ticks : Cardinal
2697: Function Time : TDateTime
2698: function Time: TDateTime;
2699: function TimeGetTime: int64;
2700: Function TimeOf( const AValue : TDateTime) : TDateTime

```

```

2701: function TimeSeparator: char;
2702: functionTimeStampToDateTIme(const TimeStamp: TTimeStamp): TDateTime
2703: FunctionTimeStampToMSecs( TimeStamp : TTimeStamp ) : Comp
2704: functionTimeStampToMSecs(const TimeStamp: TTimeStamp): Comp
2705: FunctionTimeToStr( DateTime : TDateTime ) : string;
2706: functionTimeToStr(const DateTime: TDateTime): string;
2707: FunctionTimeZoneBias : TDateTime
2708: FunctionToCommon( const AValue : Double ) : Double
2709: functionToCommon(const AValue: Double): Double;
2710: FunctionToday : TDateTime
2711: FunctionToggleBit( const Value : Byte; const Bit : TBitRange ) : Byte;
2712: FunctionToggleBit1( const Value : Shortint; const Bit : TBitRange ) : Shortint;
2713: FunctionToggleBit2( const Value : Smallint; const Bit : TBitRange ) : Smallint;
2714: FunctionToggleBit3( const Value : Word; const Bit : TBitRange ) : Word;
2715: FunctionToggleBit4( const Value : Cardinal; const Bit : TBitRange ) : Cardinal;
2716: FunctionToggleBit5( const Value : Integer; const Bit : TBitRange ) : Integer;
2717: FunctionToggleBit6( const Value : Int64; const Bit : TBitRange ) : Int64;
2718: functionTokenComponentIdent:string
2719: FunctionTokenFloat : Extended
2720: functionTokenFloat:Extended
2721: FunctionTokenInt : Longint
2722: functionTokenInt:LongInt
2723: FunctionTokenString : string
2724: functionTokenString:String
2725: FunctionTokenSymbolIs( const S : string ) : Boolean
2726: functionTokenSymbolIs(S:String):Boolean
2727: FunctionTomorrow : TDateTime
2728: FunctionToRightOf( const pc : TControl; piSpace : Integer ) : Integer
2729: FunctionToString : string
2730: FunctionTotalVariance( const Data : array of Double ) : Extended
2731: FunctionTrace2( AURL : string ) : string;
2732: FunctionTrackMenu( Button : TToolButton ) : Boolean
2733: FunctionTRANSLATE( SRC, DEST : PCHAR; TOOEM : BOOLEAN ) : INTEGER
2734: FunctionTranslateURI( const URI : string ) : string
2735: FunctionTranslationMatchesLanguages( Exact : Boolean ) : Boolean
2736: FunctionTransparentStretchBlt( DstDC : HDC; DstX,DstY,DstH:Integer;SrcDC:HDC;SrcX,SrcY,SrcW,
  SrcH:Integer;MaskDC : HDC; MaskX, MaskY : Integer ) : Boolean
2737: FunctionTrim( S : string ) : string;
2738: FunctionTrim( S : WideString ) : WideString;
2739: FunctionTrim(s : AnyString) : AnyString;
2740: FunctionTrimAllOf( ATrim, AText : String ) : String
2741: FunctionTrimLeft( S : string ) : string;
2742: FunctionTrimLeft( S : WideString ) : WideString;
2743: functionTrimLeft(const S: string): string
2744: FunctionTrimRight( S : string ) : string;
2745: FunctionTrimRight( S : WideString ) : WideString;
2746: functionTrimRight(const S: string): string
2747: functionTrueBoolStrs: array of string
2748: FunctionTrunc(e : Extended) : Longint;
2749: FunctionTrunc64(e: extended): Int64;
2750: FunctionTruncPower( const Base, Exponent : Float ) : Float
2751: FunctionTryConvTypeToFamily( const AFrom : TConvType; out AFamily : TConvFamily ) : Boolean;
2752: FunctionTryConvTypeToFamily1( const AType : TConvType; out AFamily : TConvFamily ) : Boolean;
2753: functionTryEncodeDate(Year, Month, Day: Word; var Date: TDateTime): Boolean;
2754: FunctionTryEncodeDateDay( const AYear, ADayOfYear : Word; out AValue : TDateTime ) : Boolean
2755: FunctionTryEncodeDateMonthWeek( const AY,AMonth,AWeekOfMonth,ADayOfWeek:Word;var AValue:TDateTime): Boolean
2756: FunctionTryEncodeDateTime( const AYear,AMonth,ADay,AMin,Asec,AMilliSecond:Word; out
  AValue:TDateTime):Boolean
2757: FunctionTryEncodeDateWeek( const AY,AWeekOfYear:Word;out AValue:TDateTime;const ADayOfWeek:Word): Boolean
2758: FunctionTryEncodeDayOfWeekInMonth( const AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word;out
  AVal:TDateTime):Bool
2759: functionTryEncodeTime(Hour, Min, Sec, MSec: Word; var Time: TDateTime): Boolean;
2760: FunctionTryFloatToDateTIme( Value : Extended; AResult : TDateTime ) : Boolean
2761: FunctionTryJulianToDateTIme( const AValue : Double; out ADateTime : TDateTime ) : Boolean
2762: FunctionTryLock : Boolean
2763: FunctionTryModifiedJulianDateToDateTIme( const AValue : Double; out ADateTime : TDateTime ) : Boolean
2764: FunctionTryRecodeDateTime( const AValue : TDateTime; const AYear, AMonth, ADay, AHour, AMinute, ASecond,
  AMilliSecond : Word; out AResult : TDateTime ) : Boolean
2765: FunctionTryStrToBcd( const AValue : string; var Bcd : TBcd ) : Boolean
2766: FunctionTryStrToConv( AText : string; out AValue : Double; out AType : TConvType ) : Boolean
2767: FunctionTryStrToDate( S : string; Value : TDateTime ) : Boolean;
2768: FunctionTryStrToDateTIme( S : string; Value : TDateTime ) : Boolean;
2769: FunctionTryStrToTime( S : string; Value : TDateTime ) : Boolean;
2770: FunctionTryStrToInt( const S: AnsiString; var I: Integer): Boolean;
2771: FunctionTryStrToInt64( const S: AnsiString; var I: Int64): Boolean;
2772: FunctionTwoByteToWord( AByte1, AByte2 : Byte) : Word
2773: FunctionTwoCharToWord( AChar1, AChar2 : Char ) : Word
2774: FunctionTwoToY( const Y : Float ) : Float
2775: FunctionUCS4StringToWideString( const S : UCS4String ) : WideString
2776: FunctionUIDL( const ADest : TStrings; const AMsgNum : Integer ) : Boolean
2777: functionUnassigned: Variant;
2778: FunctionUndoLastChange( FollowChange : Boolean ) : Boolean
2779: functionUniCodeToStr(Value: string): string;
2780: FunctionUnionRect( out Rect : TRect; const R1, R2 : TRect ) : Boolean
2781: functionUnionRect(out Rect: TRect; const R1, R2: TRect): Boolean)
2782: FunctionUnixDateTImeToDelphiDateTIme( UnixDateTIme : Cardinal ) : TDateTime
2783: FunctionUnixPathToDosPath( const Path : string ) : string
2784: FunctionUnixToDateTIme( const AValue : Int64 ) : TDateTime
2785: functionUnixToDateTIme(U: Int64): TDateTime;

```

```

2786: Function UnlockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
2787: Function UnlockResource( ResData : HGLOBAL) : LongBool
2788: Function UnlockVolume( var Handle : THandle) : Boolean
2789: Function UnMaskString( Mask, Value : String) : String
2790: function UpCase(ch : Char) : Char;
2791: Function UpCaseFirst( const AStr : string) : string
2792: Function UpCaseFirstWord( const AStr : string) : string
2793: Function UpdateAction( Action : TBasicAction) : Boolean
2794: Function UpdateKind : TUpdateKind
2795: Function UPDATESTATUS : TUPDATESTATUS
2796: Function UpperCase( S : string) : string
2797: Function Uppercase(s : AnyString) : AnyString;
2798: Function URLDecode( ASrc : string) : string
2799: Function URLEncode( const ASrc : string) : string
2800: Function UseRightToLeftAlignment : Boolean
2801: Function UseRightToLeftAlignmentForField( const AField : TField; Alignment : TAlignment) : Boolean
2802: Function UseRightToLeftReading : Boolean
2803: Function UTF8CharLength( Lead : Char) : Integer
2804: Function UTF8CharSize( Lead : Char) : Integer
2805: Function UTF8Decode( const S : UTF8String) : WideString
2806: Function UTF8Encode( const WS : WideString) : UTF8String
2807: Function UTF8LowerCase( const S : UTF8String) : UTF8String
2808: Function Utf8ToAnsi( const S : UTF8String) : string
2809: Function Utf8ToAnsiEx( const S : UTF8String; const cp : integer) : string
2810: Function UTF8UpperCase( const S : UTF8String) : UTF8String
2811: Function ValidFieldIndex( FieldIndex : Integer) : Boolean
2812: Function ValidParentForm(control: TControl): TForm
2813: Function Value : Variant
2814: Function ValueExists( const Section, Ident : string) : Boolean
2815: Function ValueOf( const Key : string) : Integer
2816: Function ValueInSet(AValue: Variant; ASet: Variant): Boolean;
2817: Function VALUEOFKEY( const AKEY : VARIANT) : VARIANT
2818: Function VarArrayFromStrings( Strings : TStrings) : Variant
2819: Function VarArrayFromWideStrings( Strings : TWideStrings) : Variant
2820: Function VarArrayGet(var S : Variant; I : Integer) : Variant;
2821: Function VarFMTBcd : TVarType
2822: Function VarFMTBcdCreate1 : Variant;
2823: Function VarFMTBcdCreate2( const AValue : string; Precision, Scale : Word) : Variant;
2824: Function VarFMTBcdCreate3( const AValue : Double; Precision : Word; Scale : Word) : Variant;
2825: Function VarFMTBcdCreate4( const ABcd : TBcd) : Variant;
2826: Function Variance( const Data : array of Double) : Extended
2827: Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
2828: Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
2829: Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
2830: Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
2831: Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
2832: Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
2833: Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
2834: Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
2835: Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
2836: Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
2837: Function VariantNeg( const V1 : Variant) : Variant
2838: Function VariantNot( const V1 : Variant) : Variant
2839: Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
2840: Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
2841: Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
2842: Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
2843: Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
2844: function VarIsEmpty(const V: Variant): Boolean;
2845: Function VarIsFMTBcd( const AValue : Variant) : Boolean;
2846: function VarIsNull(const V: Variant): Boolean;
2847: Function VarToBcd( const AValue : Variant) : TBcd
2848: function VarType(const V: Variant): TVarType;
2849: Function VarType( const V : Variant) : TVarType
2850: Function VarAsType( const V : Variant; AVarType : TVarType) : Variant
2851: Function VarIsType( const V : Variant; AVarType : TVarType) : Boolean;
2852: Function VarIsType1( const V : Variant; const AVarTypes : array of TVarType) : Boolean;
2853: Function VarIsByRef( const V : Variant) : Boolean
2854: Function VarIsEmpty( const V : Variant) : Boolean
2855: Procedure VarCheckEmpty( const V : Variant)
2856: Function VarIsNull( const V : Variant) : Boolean
2857: Function VarIsClear( const V : Variant) : Boolean
2858: Function VarIsCustom( const V : Variant) : Boolean
2859: Function VarIsOrdinal( const V : Variant) : Boolean
2860: Function VarIsFloat( const V : Variant) : Boolean
2861: Function VarIsNumeric( const V : Variant) : Boolean
2862: Function VarIsStr( const V : Variant) : Boolean
2863: Function VarToStr( const V : Variant) : string
2864: Function VarToStrDef( const V : Variant; const ADefault : string) : string
2865: Function VarToWideStr( const V : Variant) : WideString
2866: Function VarToWideStrDef( const V : Variant; const ADefault : WideString) : WideString
2867: Function VarToDateTime( const V : Variant) : TDateTime
2868: Function VarFromDateTime( const DateTime : TDateTime) : Variant
2869: Function VarInRange( const AValue, AMin, AMax : Variant) : Boolean
2870: Function VarEnsureRange( const AValue, AMin, AMax : Variant) : Variant
2871: TVariantRelationship', '( vrEqual, vrLessThan, vrGreater Than, vrNotEqual )'
2872: Function VarSameValue( const A, B : Variant) : Boolean
2873: Function VarCompareValue( const A, B : Variant) : TVariantRelationship
2874: Function VarIsEmptyParam( const V : Variant) : Boolean

```

```

2875: Function VarIsErrorHandler( const V : Variant; out AResult : HRESULT) : Boolean;
2876: Function VarIsErrorL( const V : Variant) : Boolean;
2877: Function VarAsError( AResult : HRESULT) : Variant;
2878: Procedure VarCopyNoInd( var Dest : Variant; const Source : Variant);
2879: Function VarIsArray( const A : Variant) : Boolean;
2880: Function VarIsArrayL( const A : Variant; AResolveByRef : Boolean) : Boolean;
2881: Function VarArrayCreate( const Bounds : array of Integer; AVarType : TVarType) : Variant;
2882: Function VarArrayOf( const Values : array of Variant) : Variant;
2883: Function VarArrayRef( const A : Variant) : Variant;
2884: Function VarTypeIsValidArrayType( const AVarType : TVarType) : Boolean;
2885: Function VarTypeIsValidElementType( const AVarType : TVarType) : Boolean;
2886: Function VarArrayDimCount( const A : Variant) : Integer;
2887: Function VarArrayLowBound( const A : Variant; Dim : Integer) : Integer;
2888: Function VarArrayHighBound( const A : Variant; Dim : Integer) : Integer;
2889: Function VarArrayLock( const A : Variant) : __Pointer;
2890: Procedure VarArrayUnlock( const A : Variant);
2891: Function VarArrayGet( const A : Variant; const Indices : array of Integer) : Variant;
2892: Procedure VarArrayPut( var A : Variant; const Value : Variant; const Indices : array of Integer);
2893: Procedure DynArrayToVariant( var V : Variant; const DynArray : __Pointer; TypeInfo : __Pointer);
2894: Procedure DynArrayFromVariant( var DynArray : __Pointer; const V : Variant; TypeInfo : __Pointer);
2895: Function Unassigned : Variant;
2896: Function Null : Variant;
2897: Function VectorAdd( const V1, V2 : TFlopPoint) : TFlopPoint;
2898: function VectorAdd(const V1,V2: TFlopPoint): TFlopPoint;
2899: Function VectorDot( const V1, V2 : TFlopPoint) : Double;
2900: function VectorDot(const V1,V2: TFlopPoint): Double;
2901: Function VectorLengthSqr( const V : TFlopPoint) : Double;
2902: function VectorLengthSqr(const V: TFlopPoint): Double;
2903: Function VectorMult( const V : TFlopPoint; const s : Double) : TFlopPoint;
2904: function VectorMult(const V: TFlopPoint; const s: Double): TFlopPoint;
2905: Function VectorSubtract( const V1, V2 : TFlopPoint) : TFlopPoint;
2906: function VectorSubtract(const V1,V2: TFlopPoint): TFlopPoint;
2907: Function Verify( AUserName : String) : String;
2908: Function Versine( X : Float) : Float;
2909: function VersionCheck: boolean;
2910: function VersionCheckAct: string;
2911: Function VersionLanguageId( const LangIdRec : TLangIdRec) : string;
2912: Function VersionLanguageName( const LangId : Word) : string;
2913: Function VersionResourceAvailable( const FileName : string) : Boolean;
2914: Function Visible : Boolean;
2915: function VolumeID(DriveChar: Char): string;
2916: Function WaitFor( const AString : string) : string;
2917: Function WaitFor( const TimeOut : Cardinal) : TJclWaitResult;
2918: Function WaitFor1 : TWaitResult;
2919: Function WaitForData( Timeout : Longint) : Boolean;
2920: Function WebColorNameToColor( WebColorName : string) : TColor;
2921: Function WebColorStrToColor( WebColor : string) : TColor;
2922: Function WebColorToRGB( WebColor : Integer) : Integer;
2923: Function wGet(aURL, afile: string): boolean;
2924: Function wGet2(aURL, afile: string): boolean; //without file open;
2925: Function WebGet(aURL, afile: string): boolean;
2926: Function WebExists: boolean; //alias to isinternet;
2927: Function WeekOf( const AValue : TDateTime) : Word;
2928: Function WeekOfTheMonth( const AValue : TDateTime) : Word;
2929: Function WeekOfTheMonth1( const AValue : TDateTime; var AYear, AMonth : Word) : Word;
2930: Function WeekOfTheYear( const AValue : TDateTime) : Word;
2931: Function WeekOfTheYear1( const AValue : TDateTime; var AYear : Word) : Word;
2932: Function WeeksBetween( const ANow, AThen : TDateTime) : Integer;
2933: Function WeeksInAYear( const AYear : Word) : Word;
2934: Function WeeksInYear( const AValue : TDateTime) : Word;
2935: Function WeekSpan( const ANow, AThen : TDateTime) : Double;
2936: Function WideAdjustLineBreaks( const S : WideString; Style : TTextLineBreakStyle) : WideString;
2937: Function WideCat( const x, y : WideString) : WideString;
2938: Function WideCompareStr( S1, S2 : WideString) : Integer;
2939: function WideCompareStr(const S1: WideString; const S2: WideString): Integer;
2940: Function WideCompareText( S1, S2 : WideString) : Integer;
2941: function WideCompareText(const S1: WideString; const S2: WideString): Integer;
2942: Function WideCopy( const src : WideString; index, count : Integer) : WideString;
2943: Function WideDequotedStr( const S : WideString; AQuote : WideChar) : WideString;
2944: Function WideEqual( const x, y : WideString) : Boolean;
2945: function WideFormat(const Format: WideString; const Args: array of const): WideString;
2946: Function WideGreater( const x, y : WideString) : Boolean;
2947: Function WideLength( const src : WideString) : Integer;
2948: Function WideLess( const x, y : WideString) : Boolean;
2949: Function WideLowerCase( S : WideString) : WideString;
2950: function WidLowerCase(const S: WideString): WideString;
2951: Function WidePos( const src, sub : WideString) : Integer;
2952: Function WideQuotedStr( const S : WideString; Quote : WideChar) : WideString;
2953: Function WideReplaceStr( const AText, AFromText, AToText : WideString) : WideString;
2954: Function WideReplaceText( const AText, AFromText, AToText : WideString) : WideString;
2955: Function WideSameStr( S1, S2 : WideString) : Boolean;
2956: function WideSameStr(const S1: WideString; const S2: WideString): Boolean;
2957: Function WideSameText( S1, S2 : WideString) : Boolean;
2958: function WideSameText(const S1: WideString; const S2: WideString): Boolean;
2959: Function WideStringReplace(const S,OldPattern, NewPattern: Widestring; Flags: TReplaceFlags): Widestring;
2960: Function WideStringToUCS4String( const S : WideString) : UCS4String;
2961: Function WidUpperCase( S : WideString) : WideString;
2962: Function Win32BackupFile( const FileName : string; Move : Boolean) : Boolean;
2963: function Win32Check(RetVal: boolean): boolean;

```

```

2964: Function Win32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
2965: Function Win32RestoreFile( const FileName : string) : Boolean
2966: Function Win32Type : TIIdWin32Type
2967: Function WinColor( const Color32 : TColor32) : TColor
2968: function winexec(FileName: pchar; showCmd: integer): integer;
2969: Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
2970: Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
2971: Function WithinPastDays( const ANow, AThen : TDateTime; const ADays : Integer) : Boolean
2972: Function WithinPastHours( const ANow, AThen : TDateTime; const AHours : Int64) : Boolean
2973: Function WithinPastMilliseconds( const ANow, AThen : TDateTime; const AMilliseconds : Int64) : Boolean
2974: Function WithinPastMinutes( const ANow, AThen : TDateTime; const AMinutes : Int64) : Boolean
2975: Function WithinPastMonths( const ANow, AThen : TDateTime; const AMonths : Integer) : Boolean
2976: Function WithinPastSeconds( const ANow, AThen : TDateTime; const ASeconds : Int64) : Boolean
2977: Function WithinPastYears( const ANow, AThen : TDateTime; const AYears : Integer) : Boolean
2978: Function WNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PChar) : DWORD
2980: Function WordToStr( const Value : Word) : String
2981: Function WordGridFormatIdent( const Ident : string; var Value : Longint) : Boolean
2982: Function IntToWordGridFormatIdent( Value : Longint; var Ident : string) : Boolean
2983: Procedure GetWordGridFormatValues( Proc : TGetStrProc)
2984: Function WorkArea : Integer
2985: Function WrapText( Line : string; MaxCol : Integer) : string;
2986: Function WrapText2( Line, BreakStr : string; BreakChars : TSysCharSet; MaxCol : Integer) : string;
2987: Function Write( pv : Pointer; cb : Longint; pcbWritten : PLongint) : HResult
2988: function Write(Buffer:String;Count:LongInt):LongInt
2989: Function WriteClient( var Buffer, Count : Integer) : Integer
2990: Function WriteFile( const AFile : string; const AEnableTransferFile : Boolean) : Cardinal
2991: Function WriteHeaders( StatusCode : Integer; const ReasonString, Headers : string) : Boolean
2992: Function WriteString( const AString : string) : Boolean
2993: Function WStrGet(var S : AnyString; I : Integer) : WideChar;
2994: Function vwsprintf( Output : PChar; Format : PChar; arglist : va_list) : Integer
2995: Function wsprintf( Output : PChar; Format : PChar) : Integer
2996: Function XmlDateTimeToStr( const XmlDateTime : string; const Format : string) : string
2997: Function XmlTimeToStr( const XmlTime : string; const Format : string) : string
2998: Function XorDecode( const Key, Source : string) : string
2999: Function XorEncode( const Key, Source : string) : string
3000: Function XorString( const Key, Src : ShortString) : ShortString
3001: Function Yield : Bool
3002: Function Yearof( const AValue : TDateTime) : Word
3003: Function YearsBetween( const ANow, AThen : TDateTime) : Integer
3004: Function YearSpan( const ANow, AThen : TDateTime) : Double
3005: Function Yesterday : TDateTime
3006: Function YesNoDialog(const ACaption, AMsg: string): boolean;
3007: Function( const Name : string; Proc : TUserFunction)
3008: Function using Special_Scholz from 3.8.5.0
3009: Function TimeToFloat(value:Extended):Extended; // Normalstunden --> Industriestunden
3010: Function FloatToTime(value:Extended):Extended; // Industriestunden --> Normalstunden
3011: Function FloatToTime2Dec(value:Extended):Extended;
3012: Function MinToStd(value:Extended):Extended;
3013: Function MinToStdAsString(value:Extended):String;
3014: Function RoundFloatToStr(zahl:Extended; decimals:integer):String;
3015: Function RoundFloat(zahl:Extended; decimals:integer):Extended;
3016: Function Round2Dec (zahl:Extended):Extended;
3017: Function GetAngle(x,y:Extended):Double;
3018: Function AddAngle(a1,a2:Double):Double;
3019:
3020: ****
3021: unit UPSI_StText;
3022: ****
3023: Function TextSeek( var F : TextFile; Target : LongInt) : Boolean
3024: Function TextFileSize( var F : TextFile) : LongInt
3025: Function TextPos( var F : TextFile) : LongInt
3026: Function TextFlush( var F : TextFile) : Boolean
3027:
3028: ****
3029: from JvVCLUtils;
3030: ****
3031: { Windows resources (bitmaps and icons) VCL-oriented routines }
3032: procedure DrawBitmapTransparent(Dest:TCanvas;DstX,DstY:Integer;Bitmap:TBitmap;TransparentColor:TColor);
3033: procedure DrawBitmapRectTransparent(Dest : TCanvas; DstX, DstY: Integer; SrcRect: TRect; Bitmap: TBitmap; TransparentColor: TColor);
3034: procedure StretchBitmapRectTransparent(Dest : TCanvas; DstX, DstY, DstW,DstH: Integer; SrcRect: TRect; Bitmap: TBitmap; TransparentColor: TColor);
3035: function MakeBitmap(ResID: PChar): TBitmap;
3036: function MakeBitmapID(ResID: Word): TBitmap;
3037: function MakeModuleBitmap(Module: THandle; ResID: PChar): TBitmap;
3038: function CreateTwoColorsBrushPattern(Color1, Color2: TColor): TBitmap;
3039: function CreateDisabledBitmap_NewStyle(FOriginal: TBitmap; BackColor: TColor): TBitmap;
3040: function CreateDisabledBitmapEx(FOriginal: TBitmap; OutlineColor, BackColor, HighlightColor, ShadowColor: TColor; DrawHighlight: Boolean): TBitmap;
3041: function CreateDisabledBitmap(FOriginal: TBitmap; OutlineColor: TColor): TBitmap;
3043: function ChangeBitmapColor(Bitmap: TBitmap; Color, NewColor: TColor): TBitmap;
3044: procedure AssignBitmapCell(Source: TGraphic; Dest: TBitmap; Cols, Rows,Index: Integer);
3045: {$IFDEF WIN32}
3046: procedure ImageListDrawDisabled(Images: TImageList; Canvas: TCanvas;
3047: X, Y, Index: Integer; HighlightColor, GrayColor: TColor; DrawHighlight: Boolean);
3048: {$ENDIF}
3049: function MakeIcon(ResID: PChar): TIcon;
3050: function MakeIconID(ResID: Word): TIcon;

```

```

3051: function MakeModuleIcon(Module: THandle; ResID: PChar): TIcon;
3052: function CreateBitmapFromIcon(Icon: TIcon; BackColor: TColor): TBitmap;
3053: {$IFDEF WIN32}
3054: function CreateIconFromBitmap(Bitmap: TBitmap; TransparentColor: TColor): TIcon;
3055: {$ENDIF}
3056: { Service routines }
3057: procedure NotImplemented;
3058: procedure ResourceNotFound(ResID: PChar);
3059: function PointInRect(const P: TPoint; const R: TRect): Boolean;
3060: function PointInPolyRgn(const P: TPoint; const Points: array of TPoint): Boolean;
3061: function PaletteColor(Color: TColor): Longint;
3062: function WidthOf(R: TRect): Integer;
3063: function HeightOf(R: TRect): Integer;
3064: procedure PaintInverseRect(const RectOrg, RectEnd: TPoint);
3065: procedure DrawInvertFrame(ScreenRect: TRect; Width: Integer);
3066: procedure CopyParentImage(Control: TControl; Dest: TCanvas);
3067: procedure Delay(MSecs: Longint);
3068: procedure CenterControl(Control: TControl);
3069: Function PaletteEntries( Palette : HPALETTE ) : Integer
3070: Function WindowClassName( Wnd : HWND ) : string
3071: Function ScreenWorkArea : TRect
3072: Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer )
3073: Procedure SwitchToWindow( Wnd : HWND; Restore : Boolean )
3074: Procedure ActivateWindow( Wnd : HWND )
3075: Procedure ShowWinNoAnimate( Handle : HWND; CmdShow : Integer )
3076: Procedure CenterWindow( Wnd : HWND )
3077: Procedure ShadeRect( DC : HDC; const Rect : TRect )
3078: Procedure KillMessage( Wnd : HWND; Msg : Cardinal )
3079: Function DialogsToPixelsX( Dlgs : Word ) : Word
3080: Function DialogsToPixelsY( Dlgs : Word ) : Word
3081: Function PixelsToDialogsX( Pixs : Word ) : Word
3082: Function PixelsToDialogsY( Pixs : Word ) : Word
3083: {$IFDEF WIN32}
3084: procedure ShowMDIClientEdge(ClientHandle: THandle; ShowEdge: Boolean);
3085: function MakeVariant(const Values: array of Variant): Variant;
3086: {$ENDIF}
3087: function CreateRotatedFont(Font: TFont; Angle: Integer): HFONT;
3088: function MsgBox(const Caption, Text: string; Flags: Integer): Integer;
3089: function MsgDlg(const Msg:string; AType:TMsgDlgType; AButtons: TMsgDlgButtons; HelpCtx: Longint): Word;
3090: {$IFDEF CBUILER}
3091: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): HWND;
3092: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
3093: {$ELSE}
3094: function FindPrevInstance(const MainFormClass, ATitle: string): HWND;
3095: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
3096: {$ENDIF CBUILER}
3097: function IsForegroundTask: Boolean;
3098: procedure MergeForm(AControl: TWinControl; AForm: TForm; Align: TAlign; Show: Boolean);
3099: function GetAveCharSize(Canvas: TCanvas): TPoint;
3100: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
3101: procedure FreeUnusedOLE;
3102: procedure Beep;
3103: function GetWindowsVersion: string;
3104: function LoadDLL(const LibName: string): THandle;
3105: function RegisterServer(const ModuleName: string): Boolean;
3106: {$IFDEF WIN32}
3107: function IsLibrary: Boolean;
3108: {$ENDIF}
3109: { Gradient filling routine }
3110: type TFillDirection = (fdTopToBottom, fdBottomToTop, fdLeftToRight, fdRightToLeft);
3111: procedure GradientFillRect(Canvas: TCanvas; ARect: TRect; StartColor, EndColor: TColor; Direction: TFillDirection; Colors: Byte);
3112: { String routines }
3113: function GetEnvVar(const VarName: string): string;
3114: function AnsiUpperFirstChar(const S: string): string;
3115: function StringToPChar(var S: string): PChar;
3116: function StrAlloc(const S: string): PChar;
3117: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
3118: function DropT(const S: string): string;
3119: { Memory routines }
3120: function AllocMemo(Size: Longint): Pointer;
3121: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
3122: procedure FreeMemo(var fpBlock: Pointer);
3123: function GetMemosize(fpBlock: Pointer): Longint;
3124: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
3125: {$IFDEF COMPILERS_UP}
3126: procedure FreeAndNil(var Obj);
3127: {$ENDIF}
3128: // from PNGloader
3129: Function OptimizeForPNG(Image:TLinearBitmap;QuantizationSteps:Integer;TransparentColor:TColor):Integer
3130: Procedure TransformRGB2LCOO( Image : TLinearBitmap)
3131: Procedure TransformLCO2RGB( Image : TLinearBitmap)
3132: Procedure SortPalette( const Pal : TPalette; var ColorMap : TColorMap)
3133: Function DrawButtonFace( Canvas : TCanvas; const Client : TRect; BevelWidth : Integer; Style : TButtonStyle; IsRounded, IsDown, IsFocused : Boolean) : TRect //TButtons
3134: Function IsAnAllResult( const AModalResult : TModalResult ) : Boolean
3135: Function InitWndProc( HWindow : HWnd; Message, WParam : Longint; LParam : Longint ) : Longint
3136: AddConstantN('CTL3D_ALL','LongWord').SetUInt($FFFF);
3137: //Procedure ChangeBiDiModeAlignment( var Alignment : TAlignment)

```

```

3138: //Function SendAppMessage( Msg : Cardinal; WParam, LParam : Longint ) : Longint
3139: //Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3140: Procedure SetIMEMode( hWnd : HWND; Mode : TImeMode )
3141: Procedure SetIMEName( Name : TIMEName )
3142: Function Win32NLSEnableIME( hWnd : HWND; Enable : Boolean ) : Boolean
3143: Function Imm32GetContext( hWnd : HWND ) : HIMC
3144: Function Imm32ReleaseContext( hWnd : HWND; hIMC : HIMC ) : Boolean
3145: Function Imm32GetConversionStatus( hIMC : HIMC; var Conversion, Sentence : longword ) : Boolean
3146: Function Imm32SetConversionStatus( hIMC : HIMC; Conversion, Sentence : longword ) : Boolean
3147: Function Imm32SetOpenStatus( hIMC : HIMC; fOpen : Boolean ) : Boolean
3148: // Function Imm32SetCompositionWindow( hIMC : HIMC; lpCompForm : PCOMPOSITIONFORM ) : Boolean
3149: // Function Imm32SetCompositionFont( hIMC : HIMC; lpLogFont : PLOGFONTA ) : Boolean
3150: Function Imm32GetCompositionString(hIMC:HIMC;dWord1:longword;lpBuf:string;dwBufLen:longint):Longint
3151: Function Imm32ISIME( hKL : longword ) : Boolean
3152: Function Imm32NotifyIME( hIMC : HIMC; dwAction, dwIndex, dwValue:longword):Boolean
3153: Procedure DragDone( Drop : Boolean )
3154:
3155:
3156: //*****added from jvvcutils
3157: function CanvasMaxTextHeight(Canvas: TCanvas): Integer;
3158: function ReplaceComponentReference(This,NewReference:TComponent;var VarReference:TComponent):Boolean;
3159: procedure DrawLine(Canvas: TCanvas; X, Y, X2, Y2: Integer);
3160: function IsPositiveResult(Value: TModalResult): Boolean;
3161: function IsNegativeResult(Value: TModalResult): Boolean;
3162: function IsAbortResult(const Value: TModalResult): Boolean;
3163: function StripAllFromResult(const Value: TModalResult): TModalResult;
3164: // returns either BrightColor or DarkColor depending on the luminance of AColor
3165: // This function gives the same result (AFAIK) as the function used in Windows to
3166: // calculate the desktop icon text color based on the desktop background color
3167: function SelectColorByLuminance(AColor, DarkColor, BrightColor: TColor): TColor;
3168: type TJvHTMLCalcType = (htmlShow, htmlCalcWidth, htmlCalcHeight, htmlHyperLink);
3169:
3170: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3171:   const State: TOwnerDrawState; const Text: string; var Width: Integer;
3172:   CalcType: TJvHTMLCalcType; MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3173:   var LinkName: string; Scale: Integer = 100); overload;
3174: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3175:   const State: TOwnerDrawState; const Text: string; var Width, Height: Integer;
3176:   CalcType: TJvHTMLCalcType; MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3177:   var LinkName: string; Scale: Integer = 100); overload;
3178: function HTMLDrawText(Canvas: TCanvas; Rect: TRect;
3179:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): string;
3180: function HTMLDrawTextHL(Canvas: TCanvas; Rect: TRect;
3181:   const State: TOwnerDrawState; const Text: string; MouseX, MouseY: Integer;
3182:   Scale: Integer = 100): string;
3183: function HTMLPlainText(const Text: string): string;
3184: function HTMLTextExtent(Canvas: TCanvas; Rect: TRect;
3185:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): TSize;
3186: function HTMLTextWidth(Canvas: TCanvas; Rect: TRect;
3187:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): Integer;
3188: function HTMLTextHeight(Canvas: TCanvas; const Text: string; Scale: Integer = 100): Integer;
3189: function HTMLPrepareText(const Text: string): string;
3190:
3191: ***** uPSI_JvAppUtils;
3192: Function GetDefaultSection( Component : TComponent ) : string
3193: Procedure GetDefaultIniData( Control:TControl; var IniFileName, Section : string; UseRegistry : Boolean )
3194: Procedure GetDefaultIniData( Control : TControl; var IniFileName, Section : string )
3195: Function GetDefaultIniName : string
3196: //OnGetDefaultIniName , 'TOnGetDefaultIniName';
3197: Function GetDefaultIniKey : string
3198: Function FindForm( FormClass : TFormClass ) : TForm
3199: Function FindShowForm( FormClass : TFormClass; const Caption : string ) : TForm
3200: Function ShowDialog( FormClass : TFormClass ) : Boolean
3201: //Function InstantiateForm( FormClass : TFormClass; var Reference ) : TForm
3202: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean )
3203: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean )
3204: Procedure SaveMDIChildrenReg( MainForm : TForm; IniFile : TRegIniFile )
3205: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string )
3206: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string )
3207: Function GetUniqueFileNameInDir( const Path, FileNameMask : string ) : string
3208: Function StrToIniParam( const Str : string ) : string
3209: Function IniParamToStr( const Str : string ) : string
3210: Function IniParamReadString( IniFile : TObject; const Section, Ident, Default : string ) : string
3211: Procedure IniParamWriteString( IniFile : TObject; const Section, Ident, Value : string )
3212: Function IniParamReadInteger( IniFile : TObject; const Section, Ident : string; Default : Longint ) : Longint
3213: Procedure IniParamWriteInteger( IniFile : TObject; const Section, Ident : string; Value : Longint )
3214: Function IniParamReadBool( IniFile : TObject; const Section, Ident : string; Default : Boolean ) : Boolean
3215: Procedure IniParamWriteBool( IniFile : TObject; const Section, Ident : string; Value : Boolean )
3216: Procedure IniParamReadSections( IniFile : TObject; Strings : TStrings )
3217: Procedure IniParamEraseSection( IniFile : TObject; const Section : string )
3218: Procedure IniParamDeleteKey( IniFile : TObject; const Section, Ident : string )
3219: Procedure AppBroadcast( Msg, wParam : Longint; lParam : Longint )
3220: Procedure AppBroadcast( Msg, wParam : Word; lParam : Longint )
3221: Procedure AppTaskbarIcons( AppOnly : Boolean )
3222: Procedure InternalSaveGridLayout( Grid : TCustomGrid; IniFile : TObject; const Section : string )
3223: Procedure InternalRestoreGridLayout( Grid : TCustomGrid; IniFile : TObject; const Section : string )
3224: Procedure InternalSaveMDIChildren( MainForm : TForm; IniFile : TObject )
3225: Procedure InternalRestoreMDIChildren( MainForm : TForm; IniFile : TObject )
3226: ***** uPSI_JvDBUtils;

```

```

3227: Function CreateLocate( DataSet : TDataSet ) : TJvLocateObject
3228: Function IsDataSetEmpty( DataSet : TDataSet ) : Boolean
3229: Procedure RefreshQuery( Query : TDataSet )
3230: Function DataSetSortedSearch(DataSet:TDataSet;const Value,FieldName:string;CaseInsensitive:Bool):Boolean
3231: Function DataSetSectionName( DataSet : TDataSet ) : string
3232: Procedure InternalSaveFields( DataSet : TDataSet; IniFile : TObject; const Section : string )
3233: Procedure InternalRestoreFields(DataSet:TDataSet;IniFile:TObject;const Section:string;RestoreVisible:Bool)
3234: Function DataSetLocateThrough(DataSet:TDataSet; const KeyFields: string; const KeyValues: Variant;
  Options: TLocateOptions ) : Boolean
3235: Procedure SaveFields( DataSet : TDataSet; IniFile : TIniFile )
3236: Procedure RestoreFields( DataSet : TDataSet; IniFile : TIniFile; RestoreVisible : Boolean )
3237: Procedure AssignRecord( Source, Dest : TDataSet; ByName : Boolean )
3238: Function ConfirmDelete : Boolean
3239: Procedure ConfirmDataSetCancel( DataSet : TDataSet )
3240: Procedure CheckRequiredField( Field : TField )
3241: Procedure CheckRequiredFields( const Fields : array of TField )
3242: Function DateToSQL( Value : TDateTime ) : string
3243: Function FormatSQLDateRange( Date1, Date2 : TDateTime; const FieldName : string ) : string
3244: Function FormatSQLDateRangeEx( Date1, Date2 : TDateTime; const FieldName : string ) : string
3245: Function FormatSQLNumericRange(const FieldName:string;LowVal,HighVal,LowEmpty,
  HighEmpty:Double;Inclusive:Bool):string
3246: Function StrMaskSQL( const Value : string ) : string
3247: Function FormatSQLCondition(const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool):string
3248: Function FormatAnsSQLCondition(const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool):string
3249: Procedure _DBError( const Msg : string )
3250: Const ('TrueExpr','String '0=0
3251: Const ('sdfStandard16','String '''''mm''/''dd''/''yyyy''''')
3252: Const ('sdfStandard32','String '''''dd/mm/yyyy''''')
3253: Const ('sdfOracle','String ''TO_DATE('''dd/mm/yyyy''', ''DD/MM/YYYY'')')
3254: Const ('sdfInterbase','String ''CAST('''mm''/''dd''/''yyyy'''' AS DATE)'')
3255: Const ('sdfMSSQL','String ''CONVERT(datetime, '''mm''/''dd''/''yyyy''', 103)'')
3256: AddTypeS('Largeint', 'Longint'
3257: addTypeS('TIEFException', 'ErNoImport, erCannotImport, erInvalidType, ErInternalError, '+
3258:   'erInvalidHeader, erInvalidOpcode, erInvalidOpcodeParameter, erNoMainProc, erOutOfGlobalVarsRange, '+
3259:   'erOutOfProcRange, ErOutOfRange, erOutOfStackRange, ErTypeMismatch, erUnexpectedEof, '+
3260:   'erVersionError, ErDivideByZero, ErMathError,erCouldNotCallProc, erOutOfRecordRange, '+
3261:   'erOutOfMemory,erException,erNullPointerException,erNullVariantError,erInterfaceNotSupported,erUnsupportedError);
3262: (*-----*)
3263: procedure SIRegister_JclIniFiles(CL: TPSPascalCompiler);
3264: begin
3265:   Function JIniReadBool( const FileName, Section, Line : string ) : Boolean
3266:   Function JIniReadInteger( const FileName, Section, Line : string ) : Integer
3267:   Function JIniReadString( const FileName, Section, Line : string ) : string
3268:   Procedure JIniWriteBool( const FileName, Section, Line : string; Value : Boolean )
3269:   Procedure JIniWriteInteger( const FileName, Section, Line : string; Value : Integer )
3270:   Procedure JIniWriteString( const FileName, Section, Line, Value : string )
3271:   Procedure JIniReadStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings )
3272:   Procedure JIniWriteStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings )
3273: end;
3274:
3275: (* === compile-time registration functions === *)
3276: (*-----*)
3277: procedure SIRegister_JclDateTime(CL: TPSPascalCompiler);
3278: begin
3279:   'UnixTimeStart','LongInt'( 25569 );
3280:   Function JEncodeDate( const Year : Integer; Month, Day : Word ) : TDateTime
3281:   Procedure JDecodeDate( Date : TDateTime; var Year, Month, Day : Word );
3282:   Procedure DecodeDate1( Date : TDateTime; var Year : Integer; var Month, Day : Word );
3283:   Procedure DecodeDate2( Date : TDateTime; var Year, Month, Day : Integer );
3284:   Function CenturyOfDate( const Date : TDateTime ) : Integer
3285:   Function CenturyBaseYear( const Date : TDateTime ) : Integer
3286:   Function DayOfDate( const Date : TDateTime ) : Integer
3287:   Function MonthOfDay( const Date : TDateTime ) : Integer
3288:   Function YearOfDate( const Date : TDateTime ) : Integer
3289:   Function JDayOfTheYear( const Date : TDateTime; var Year : Integer ) : Integer
3290:   Function DayOfTheYear1( const Date : TDateTime ) : Integer
3291:   Function DayOfTheYearToDate( const Year, Day : Integer ) : TDateTime
3292:   Function HourOfTime( const Date : TDateTime ) : Integer
3293:   Function MinuteOfTime( const Date : TDateTime ) : Integer
3294:   Function SecondOfTime( const Date : TDateTime ) : Integer
3295:   Function GetISOYearNumberOfDays( const Year : Word ) : Word
3296:   Function IsISOLongYear( const Year : Word ) : Boolean;
3297:   Function IsISOLongYear1( const Date : TDateTime ) : Boolean;
3298:   Function ISODayOfWeek( const Date : TDateTime ) : Word
3299:   Function JISOWeekNumber( Date : TDateTime; var YearOfWeekNumber, WeekDay : Integer ) : Integer;
3300:   Function ISOWeekNumber1( Date : TDateTime; var YearOfWeekNumber : Integer ) : Integer;
3301:   Function ISOWeekNumber2( Date : TDateTime ) : Integer;
3302:   Function ISOWeekToDate( const Year, Week, Day : Integer ) : TDateTime
3303:   Function JIsLeapYear( const Year : Integer ) : Boolean;
3304:   Function IsLeapYear1( const Date : TDateTime ) : Boolean;
3305:   Function JDdaysInMonth( const Date : TDateTime ) : Integer
3306:   Function Make4DigitYear( Year, Pivot : Integer ) : Integer
3307:   Function JMakeYear4Digit( Year, WindowsillYear : Integer ) : Integer
3308:   Function JEasterSunday( const Year : Integer ) : TDateTime // TDosDateTime', 'Integer
3309:   Function JFormatDateTime( Form : string; Date : TDateTime ) : string
3310:   Function FATDatesEqual( const FileTime1, FileTime2 : Int64 ) : Boolean;
3311:   Function FATDatesEqual1( const FileTime1, FileTime2 : TFileTime ) : Boolean;
3312:   Function HoursToMSecs( Hours : Integer ) : Integer
3313:   Function MinutesToMSecs( Minutes : Integer ) : Integer

```

```

3314: Function SecondsToMSEcs( Seconds : Integer ) : Integer
3315: Function TimeOfDateTimeToSeconds( DateTime : TDateTime ) : Integer
3316: Function TimeOfDateTimeToMSEcs( DateTime : TDateTime ) : Integer
3317: Function DateTimeToLocalDateTime( DateTime : TDateTime ) : TDateTime
3318: Function LocalDateTimeToDateTIme( DateTime : TDateTime ) : TDateTime
3319: Function DateTimeToDosDateTime( const DateTime : TDateTime ) : TDosDateTime
3320: Function JDATimeToFileTime( DateTime : TDateTime ) : TFileTime
3321: Function JDATimeToSystemTime( DateTime : TDateTime ) : TSystemTime;
3322: Procedure DateTImeToSystemTime1( DateTime : TDateTime; var SysTime : TSystemTime );
3323: Function LocalDateTimeToFileTime( DateTime : TDateTime ) : FileTime
3324: Function DosDateTimeToDateTIme( const DosTime : TDosDateTime ) : TDateTime
3325: Function JDosDateTimeToFileTime( DosTime : TDosDateTime ) : TFileTime;
3326: Procedure DosDateTimeToFileTime1( DTH, DTL : Word; FT : TFileTime );
3327: Function DosDateTimeToSystemTime( const DosTime : TDosDateTime ) : TSystemTime
3328: Function DosDateTimeToStr( DateTime : Integer ) : string
3329: Function JFFileTimeToDateTIme( const FileTime : TFileTime ) : TDateTime
3330: Function FileTimeToLocalDateTime( const FileTime : TFileTime ) : TDateTime
3331: Function JFFileTimeToDosDateTime( const FileTime : TFileTime ) : TDosDateTime;
3332: Procedure FileTimeToDosDateTime1( const FileTime : TFileTime; var Date, Time : Word );
3333: Function JFFileTimeToSystemTime( const FileTime : TFileTime ) : TSystemTime;
3334: Procedure FileTimeToSystemTime1( const Filetime : TFileTime; var ST : TSystemTime );
3335: Function FileTimeToStr( const FileTime : TFileTime ) : string
3336: Function SystemTimeToDosDateTime( const SystemTime : TSystemTime ) : TDosDateTime
3337: Function JSystemTimeToFileTime( const SystemTime : TSystemTime ) : TFileTime;
3338: Procedure SystemTimeToFileTime1( const SystemTime : TSystemTime; FTime : TFileTime );
3339: Function SystemTimeToStr( const SystemTime : TSystemTime ) : string
3340: Function CreationDateTImeOfFile( const Sr : TSearchRec ) : TDateTime
3341: Function LastAccessDateTImeOfFile( const Sr : TSearchRec ) : TDateTime
3342: Function LastWriteDateTImeOfFile( const Sr : TSearchRec ) : TDateTime
3343: TJclUnixTime32', 'Longword
3344: Function JDATimeToUnixTime( DateTime : TDateTime ) : TJclUnixTime32
3345: Function JUnixTimeToDateTIme( const UnixTime : TJclUnixTime32 ) : TDateTime
3346: Function FileTimeToUnixTime( const AValue : TFileTime ) : TJclUnixTime32
3347: Function UnixTimeToFileTime( const AValue : TJclUnixTime32 ) : TFileTime
3348: Function JNullStamp : TTimeStamp
3349: Function JCompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp ) : Int64
3350: Function JEEqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp ) : Boolean
3351: Function JIsNullTimeStamp( const Stamp : TTimeStamp ) : Boolean
3352: Function TimeStampDOW( const Stamp : TTimeStamp ) : Integer
3353: Function FirstWeekDay( const Year, Month : Integer; var DOW : Integer ) : Integer;
3354: Function FirstWeekDay1( const Year, Month : Integer ) : Integer;
3355: Function LastWeekDay( const Year, Month : Integer; var DOW : Integer ) : Integer;
3356: Function LastWeekDay1( const Year, Month : Integer ) : Integer;
3357: Function IndexedWeekDay( const Year, Month : Integer; Index : Integer ) : Integer
3358: Function FirstWeekendDay( const Year, Month : Integer; var DOW : Integer ) : Integer;
3359: Function FirstWeekendDay1( const Year, Month : Integer ) : Integer;
3360: Function LastWeekendDay( const Year, Month : Integer; var DOW : Integer ) : Integer;
3361: Function LastWeekendDay1( const Year, Month : Integer ) : Integer;
3362: Function IndexedWeekendDay( const Year, Month : Integer; Index : Integer ) : Integer
3363: Function FirstDayOfWk( const Year, Month, DayOfWeek : Integer ) : Integer
3364: Function LastDayOfWk( const Year, Month, DayOfWk : Integer ) : Integer
3365: Function IndexedDayOfWk( const Year, Month, DayOfWk, Index : Integer ) : Integer
3366:   FindClass('TOBJECT'), 'EJclDateTimeError
3367: end;
3368:
3369: procedure SIRegister_JclMiscel2(CL: TPSPascalCompiler);
3370: begin
3371:   Function SetDisplayResolution( const XRes, YRes : DWORD ) : Longint
3372:   Function CreateDOSProcessRedirected( const CommandLine, InputFile, OutputFile : string ) : Boolean
3373:   Function WinExec32( const Cmd : string; const CmdShow : Integer ) : Boolean
3374:   Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer ) : Cardinal
3375:   Function WinExec32AndRedirectOutput(const Cmd: string; var Output: string; RawOutput:Boolean):Cardinal
3376:   TJclKillLevel', '( kNormal, kNoSignal, kTimeOut )
3377:   Function ExitWindows( ExitCode : Cardinal ) : Boolean
3378:   Function LogOffOS( KillLevel : TJclKillLevel ) : Boolean
3379:   Function PowerOffOS( KillLevel : TJclKillLevel ) : Boolean
3380:   Function ShutDownOS( KillLevel : TJclKillLevel ) : Boolean
3381:   Function RebootOS( Killlevel : TJclKillLevel ) : Boolean
3382:   Function HibernateOS( Force, DisableWakeEvents : Boolean ) : Boolean
3383:   Function SuspendOS( Force, DisableWakeEvents : Boolean ) : Boolean
3384:   Function ShutDownDialog( const DialogMessage:string;TimeOut:DWORD;Force,Reboot:Boolean ):Boolean;
3385:   Function ShutDownDialog1(const MachineName,DialogMessage:string;TimeOut:DWORD;Force,Reboot:Bool):Bool;
3386:   Function AbortShutDown : Boolean;
3387:   Function AbortShutDown1( const MachineName : string ) : Boolean;
3388:   TJclAllowedPowerOperation', '( apoHibernate, apoShutdown, apoSuspend )
3389:   TJclAllowedPowerOperations', 'set of TJclAllowedPowerOperation
3390:   Function GetAllowedPowerOperations : TJclAllowedPowerOperations
3391:   FindClass('TOBJECT'), 'EJclCreateProcessError
3392:   Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string )
3393:   Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const Environment:PChar );
3394:   // with Add(EJclCreateProcessError) do
3395: end;
3396:
3397:
3398: procedure SIRegister_JclAnsiStrings(CL: TPSPascalCompiler);
3399: begin
3400:   //AnsiSigns', 'Set').SetSet(['-', '+']);
3401:   'C1_UPPER','LongWord( $0001 );

```

```

3402: 'C1_LOWER','LongWord( $0002);
3403: 'C1_DIGIT','LongWord').SetUInt( $0004);
3404: 'C1_SPACE','LongWord').SetUInt( $0008);
3405: 'C1_PUNCT','LongWord').SetUInt( $0010);
3406: 'C1_CNTRL','LongWord').SetUInt( $0020);
3407: 'C1_BLANK','LongWord').SetUInt( $0040);
3408: 'C1_XDIGIT','LongWord').SetUInt( $0080);
3409: 'C1_ALPHA','LongWord').SetUInt( $0100);
3410: AnsiChar', 'Char
3411: Function StrIsAlpha( const S : AnsiString ) : Boolean
3412: Function StrIsAlphaNum( const S : AnsiString ) : Boolean
3413: Function StrIsAlphaNumUnderscore( const S : AnsiString ) : Boolean
3414: Function StrContainsChars( const S:AnsiString;Chars:TSysCharSet; CheckAll : Boolean ) : Boolean
3415: Function StrConsistsOfNumberChars( const S : AnsiString ) : Boolean
3416: Function StrIsDigit( const S : AnsiString ) : Boolean
3417: Function StrIsSubset( const S : AnsiString; const ValidChars : TSysCharSet ) : Boolean
3418: Function StrSame( const S1, S2 : AnsiString ) : Boolean
3419: //Function StrCenter( const S : AnsiString; L : Integer; C : AnsiChar ) : AnsiString
3420: Function StrCharPosLower( const S : AnsiString; CharPos : Integer ) : AnsiString
3421: Function StrCharPosUpper( const S : AnsiString; CharPos : Integer ) : AnsiString
3422: Function StrDoubleQuote( const S : AnsiString ) : AnsiString
3423: Function StrEnsureNoPrefix( const Prefix, Text : AnsiString ) : AnsiString
3424: Function StrEnsureNoSuffix( const Suffix, Text : AnsiString ) : AnsiString
3425: Function StrEnsurePrefix( const Prefix, Text : AnsiString ) : AnsiString
3426: Function StrEnsureSuffix( const Suffix, Text : AnsiString ) : AnsiString
3427: Function StrEscapedToString( const S : AnsiString ) : AnsiString
3428: Function JStrLower( const S : AnsiString ) : AnsiString
3429: Procedure StrLowerInPlace( var S : AnsiString )
3430: //Procedure StrLowerBuff( S : PAnsiChar )
3431: Procedure JStrMove( var Dest:AnsiString; const Source:AnsiString;const ToIndex,FromIndex,Count:Integer );
3432: Function StrPadLeft( const S : AnsiString; Len : Integer; C : AnsiChar ) : AnsiString
3433: Function StrPadRight( const S : AnsiString; Len : Integer; C : AnsiChar ) : AnsiString
3434: Function StrProper( const S : AnsiString ) : AnsiString
3435: //Procedure StrProperBuff( S : PAnsiChar )
3436: Function StrQuote( const S : AnsiString; C : AnsiChar ) : AnsiString
3437: Function StrRemoveChars( const S : AnsiString; const Chars : TSysCharSet ) : AnsiString
3438: Function StrKeepChars( const S : AnsiString; const Chars : TSysCharSet ) : AnsiString
3439: Procedure JStrReplace(var S:AnsiString; const Search, Replace : AnsiString; Flags : TReplaceFlags)
3440: Function StrReplaceChar( const S : AnsiString; const Source, Replace : AnsiChar ) : AnsiString
3441: Function StrReplaceChars( const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString
3442: Function StrReplaceButChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString;
3443: Function StrRepeat( const S : AnsiString; Count : Integer ) : AnsiString
3444: Function StrRepeatLength( const S : AnsiString; const L : Integer ) : AnsiString
3445: Function StrReverse( const S : AnsiString ) : AnsiString
3446: Procedure StrReverseInPlace( var S : AnsiString )
3447: Function StrSingleQuote( const S : AnsiString ) : AnsiString
3448: Function StrSmartCase( const S : AnsiString; Delimiters : TSysCharSet ) : AnsiString
3449: Function StrStringToEscaped( const S : AnsiString ) : AnsiString
3450: Function StrStripNonNumberChars( const S : AnsiString ) : AnsiString
3451: Function StrToHex( const Source : AnsiString ) : AnsiString
3452: Function StrTrimCharLeft( const S : AnsiString; C : AnsiChar ) : AnsiString
3453: Function StrTrimCharsLeft( const S : AnsiString; const Chars : TSysCharSet ) : AnsiString
3454: Function StrTrimCharRight( const S : AnsiString; C : AnsiChar ) : AnsiString
3455: Function StrTrimCharsRight( const S : AnsiString; const Chars : TSysCharSet ) : AnsiString
3456: Function StrTrimQuotes( const S : AnsiString ) : AnsiString
3457: Function JStrUpper( const S : AnsiString ) : AnsiString
3458: Procedure StrUpperInPlace( var S : AnsiString )
3459: //Procedure StrUpperBuff( S : PAnsiChar )
3460: Function StrOemToAnsi( const S : AnsiString ) : AnsiString
3461: Function StrAnsiToOem( const S : AnsiString ) : AnsiString
3462: Procedure StrAddRef( var S : AnsiString )
3463: Function StrAllocSize( const S : AnsiString ) : Longint
3464: Procedure StrDecRef( var S : AnsiString )
3465: //Function StrLen( S : PAnsiChar ) : Integer
3466: Function StrLength( const S : AnsiString ) : Longint
3467: Function StrRefCount( const S : AnsiString ) : Longint
3468: Procedure StrResetLength( var S : AnsiString )
3469: Function StrCharCount( const S : AnsiString; C : AnsiChar ) : Integer
3470: Function StrCharsCount( const S : AnsiString; Chars : TSysCharSet ) : Integer
3471: Function StrSCount( const S, SubS : AnsiString ) : Integer
3472: Function StrCompare( const S1, S2 : AnsiString ) : Integer
3473: Function StrCompareRange( const S1, S2 : AnsiString; const Index, Count : Integer ) : Integer
3474: //Function StrFillChar( const C : AnsiChar; Count : Integer ) : AnsiString;
3475: Function StrFillChar( const C : Char; Count : Integer ) : AnsiString;
3476: Function StrFillChar( const C: Char; Count: Integer): string';
3477: Function IntFillChar( const I: Integer; Count: Integer): string');
3478: Function ByteFillChar( const B: Byte; Count: Integer): string');
3479: Function ArrFillChar( const AC: Char; Count: Integer): TCharArray';
3480: Function ArrByteFillChar( const AB: Char; Count: Integer): TByteArray;
3481: Function StrFind( const Substr, S : AnsiString; const Index : Integer ) : Integer
3482: //Function StrHasPrefix( const S : AnsiString; const Prefixes : array of AnsiString ) : Boolean
3483: Function StrIndex( const S : AnsiString; const List : array of AnsiString ) : Integer
3484: Function StrILastPos( const SubStr, S : AnsiString ) : Integer
3485: Function StrIPos( const SubStr, S : AnsiString ) : Integer
3486: Function StrIsOneOf( const S : AnsiString; const List : array of AnsiString ) : Boolean
3487: Function StrLastPos( const SubStr, S : AnsiString ) : Integer
3488: Function StrMatch( const Substr, S : AnsiString; const Index : Integer ) : Integer
3489: Function StrMatches( const Substr, S : AnsiString; const Index : Integer ) : Boolean
3490: Function StrNIPos( const S, SubStr : AnsiString; N : Integer ) : Integer

```

```

3491: Function StrNPos( const S, SubStr : AnsiString; N : Integer ) : Integer
3492: Function StrPrefixIndex( const S : AnsiString; const Prefixes : array of AnsiString ) : Integer
3493: Function StrSearch( const Substr, S : AnsiString; const Index : Integer ) : Integer
3494: //Function StrAfter( const SubStr, S : AnsiString ) : AnsiString
3495: //Function StrBefore( const SubStr, S : AnsiString ) : AnsiString
3496: Function StrBetween( const S : AnsiString; const Start, Stop : AnsiChar ) : AnsiString
3497: Function StrChopRight( const S : AnsiString; N : Integer ) : AnsiString
3498: Function StrLeft( const S : AnsiString; Count : Integer ) : AnsiString
3499: Function StrMid( const S : AnsiString; Start, Count : Integer ) : AnsiString
3500: Function StrRestOf( const S : AnsiString; N : Integer ) : AnsiString
3501: Function StrRight( const S : AnsiString; Count : Integer ) : AnsiString
3502: Function CharEqualNoCase( const C1, C2 : AnsiChar ) : Boolean
3503: Function CharIsAlpha( const C : AnsiChar ) : Boolean
3504: Function CharIsAlphaNum( const C : AnsiChar ) : Boolean
3505: Function CharIsBlank( const C : AnsiChar ) : Boolean
3506: Function CharIsControl( const C : AnsiChar ) : Boolean
3507: Function CharIsDelete( const C : AnsiChar ) : Boolean
3508: Function CharIsDigit( const C : AnsiChar ) : Boolean
3509: Function CharIsLower( const C : AnsiChar ) : Boolean
3510: Function CharIsNumberChar( const C : AnsiChar ) : Boolean
3511: Function CharIsPrintable( const C : AnsiChar ) : Boolean
3512: Function CharIsPunctuation( const C : AnsiChar ) : Boolean
3513: Function CharIsReturn( const C : AnsiChar ) : Boolean
3514: Function CharIsSpace( const C : AnsiChar ) : Boolean
3515: Function CharIsUpper( const C : AnsiChar ) : Boolean
3516: Function CharIsWhiteSpace( const C : AnsiChar ) : Boolean
3517: Function CharType( const C : AnsiChar ) : Word
3518: Function CharHex( const C : AnsiChar ) : Byte
3519: Function CharLower( const C : AnsiChar ) : AnsiChar
3520: Function CharUpper( const C : AnsiChar ) : AnsiChar
3521: Function CharToggleCase( const C : AnsiChar ) : AnsiChar
3522: Function CharPos( const S : AnsiString; const C : AnsiChar; const Index : Integer ) : Integer
3523: Function CharLastPos( const S : AnsiString; const C : AnsiChar; const Index : Integer ) : Integer
3524: Function CharIPos( const S : AnsiString; C : AnsiChar; const Index : Integer ) : Integer
3525: Function CharReplace( var S : AnsiString; const Search, Replace : AnsiChar ) : Integer
3526: Procedure StrToStrings( S, Sep : AnsiString; const List : TStrings; const AllowEmptyString : Boolean )
3527: Procedure StrToStrings( S, Sep : AnsiString; const List : TStrings; const AllowEmptyString : Boolean )
3528: Function StringsToStr( const List:TStrings;const Sep:AnsiString;const AllowEmptyString:Bool ):AnsiString;
3529: Procedure TrimStrings( const List : TStrings; DeleteIfEmpty : Boolean )
3530: Procedure TrimStringsRight( const List : TStrings; DeleteIfEmpty : Boolean )
3531: Procedure TrimStringsLeft( const List : TStrings; DeleteIfEmpty : Boolean )
3532: Function AddStringToStrings( const S:AnsiString;Strings:TStrings; const Unique:Boolean ):Boolean
3533: Function BooleanToStr( B : Boolean ) : AnsiString
3534: Function FileToString( const FileName : AnsiString ) : AnsiString
3535: Procedure StringToFile( const FileName : AnsiString; Contents : AnsiString; Append : Boolean )
3536: Function StrToken( var S : AnsiString; Separator : AnsiChar ) : AnsiString
3537: Procedure StrTokens( const S : AnsiString; const List : TStrings )
3538: Procedure StrTokenToStrings( S : AnsiString; Separator : AnsiChar; const List : TStrings )
3539: //Function StrWord( var S : PAnsiChar; out Word : AnsiString ) : Boolean
3540: Function StrToFloatSafe( const S : AnsiString ) : Float
3541: Function StrToIntSafe( const S : AnsiString ) : Integer
3542: Procedure StrNormIndex( const StrLen : Integer; var Index : Integer; var Count : Integer );
3543: Function ArrayOf( List : TStrings ) : TDynStringArray;
3544:   EJclStringError', 'EJclError
3545: function IsClass(Address: TObject): Boolean;
3546: function IsObject(Address: TObject): Boolean;
3547: // Console Utilities
3548: //function ReadKey: Char;
3549: function IntToStrZeroPad(Value, Count: Integer): AnsiString;
3550: function JclGUIDToString(const GUID: TGUID): string;
3551: function JclStringToGUID(const S: string): TGUID;
3552:
3553: end;
3554:
3555:
3556: ***** uPSI_JvDBUtil;
3557: Procedure ExecuteSQLScript(Base:TDataBase; const Script: string; const Commit:TCommit;OnProgress:TOnProgress; const UserData : Integer)
3558: Function GetQueryResult( const DatabaseName, SQL : string ) : Variant
3559: Function GetStoredProcResult( const ADatabaseName, AStoredProcName : string; AParams : array of Variant; const AResultName : string ) : Variant
3560: //Function StrFieldDesc( Field : FLDesc ) : string
3561: Function Var2Type( V : Variant; const VarType : Integer ) : Variant
3562: Procedure CopyRecord( DataSet : TDataSet )
3563: //Procedure AddReference( Tbl : TTable; RefName : string; RefField : Word; MasterTable : string; MasterField : Word; ModOp, DelOp : RINTQual )
3564: Procedure AddMasterPassword( Table : TTable; pswd : string )
3565: Procedure PackTable( Table : TTable )
3566: Procedure PackEncryptedTable( Table : TTable; pswd : string )
3567: Function EncodeQuotes( const S : string ) : string
3568: Function Cmp( const S1, S2 : string ) : Boolean
3569: Function SubStr( const S : string; const Index : Integer; const Separator : string ) : string
3570: Function SubStrEnd( const S : string; const Index : Integer; const Separator : string ) : string
3571: Function ReplaceString( S : string; const OldPattern, NewPattern : string ) : string
3572: Procedure GetXYByPos( const S : string; const Pos : Integer; var X, Y : Integer )
3573: *****uPSI_JvJvBDEUtils;*****
3574: //JvBDEUtils
3575: Function CreateDbLocate : TJvLocateObject
3576: //Function CheckOpen( Status : DBIResult ) : Boolean

```

```

3577: Procedure FetchAllRecords( DataSet : TBDEDataSet )
3578: Function TransActive( Database : TDatabase ) : Boolean
3579: Function AsyncQrySupported( Database : TDatabase ) : Boolean
3580: Function GetQuoteChar( Database : TDatabase ) : string
3581: Procedure ExecuteQuery( const DbName, QueryText : string )
3582: Procedure ExecuteQueryEx( const SessName, DbName, QueryText : string )
3583: Function FieldLogicMap( FldType : TFieldType ) : Integer
3584: Function FieldSubtypeMap( FldType : TFieldType ) : Integer Value : string; Buffer : Pointer)
3585: Function GetAliasPath( const AliasName : string ) : string
3586: Function IsDirectory( const DatabaseName : string ) : Boolean
3587: Function GetBdeDirectory : string
3588: Function LoginToDatabase( Database : TDatabase; OnLogin : TDatabaseLoginEvent ) : Boolean
3589: Function DataSetFindValue( ADataSet : TBDEDataSet; const Value,FieldName : string ) : Boolean
3590: Function DataSetFindLike( ADataSet : TBDEDataSet; const Value,FieldName : string ) : Boolean
3591: Function DataSetRecNo( DataSet : TDataSet ) : Longint
3592: Function DataSetRecordCount( DataSet : TDataSet ) : Longint
3593: Function DataSetPositionStr( DataSet : TDataSet ) : string
3594: Procedure DataSetShowDeleted( DataSet : TBDEDataSet; Show : Boolean )
3595: Function CurrentRecordDeleted( DataSet : TBDEDataSet ) : Boolean
3596: Function IsFilterApplicable( DataSet : TDataSet ) : Boolean
3597: Function IsBookmarkStable( DataSet : TBDEDataSet ) : Boolean
3598: Procedure SetIndex( Table : TTable; const IndexFieldNames : string )
3599: Procedure RestoreIndex( Table : TTable )
3600: Procedure DeleteRange( Table : TTable; IndexFields : array of const; FieldValues : array of const )
3601: Procedure PackTable( Table : TTable )
3602: Procedure ReindexTable( Table : TTable )
3603: Procedure BdefflushBuffers
3604: Function GetNativeHandle( Database : TDatabase; Buffer : Pointer; BufSize : Integer ) : Pointer
3605: Procedure ToggleDebugLayer( Active : Boolean; const DebugFile : string )
3606: Procedure DbNotSupported
3607: Procedure ExportDataSet( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
    AsciiCharSet : string; AsciiDelimited : Boolean; MaxRecordCount : Longint )
3608: Procedure ExportDataSetEx( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
    AsciiCharSet:string;AsciiDelimited:Boolean;AsciiDelimiter,AsciiSeparator:Char;MaxRecordCount:Longint );
3609: Procedure
    ImportDataSet(Source:TBDEDataSet;DestTable:TTable;MaxRecordCount:Longint;Mappings:TStrings;Mode:TBatchMode);
3610: Procedure InitRSRUN(Database: TDatabase;const ConName:string; ConType:Integer;const ConServer:string);
3611: ****uPSI_JvDateUtil;
3612: function CurrentYear: Word;
3613: function IsLeapYear(AYear: Integer): Boolean;
3614: function DaysPerMonth(AYear, AMonth: Integer): Integer;
3615: function FirstDayOfPrevMonth: TDateTime;
3616: function LastDayOfPrevMonth: TDateTime;
3617: function FirstDayOfNextMonth: TDateTime;
3618: function ExtractDay(ADate: TDateTime): Word;
3619: function ExtractMonth(ADate: TDateTime): Word;
3620: function ExtractYear(ADate: TDateTime): Word;
3621: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
3622: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime;
3623: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
3624: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
3625: function ValidDate(ADate: TDateTime): Boolean;
3626: procedure DateDiff(Datet1, Date2: TDateTime; var Days, Months, Years: Word);
3627: function MonthsBetween(Datet1, Date2: TDateTime): Double;
3628: function DaysInPeriod(Datet1, Date2: TDateTime): Longint;
3629: { Count days between Datet1 and Date2 + 1, so if Datet1 = Date2 result = 1 }
3630: function DaysBetween(Datet1, Date2: TDateTime): Longint;
3631: { The same as previous but if Date2 < Datet1 result = 0 }
3632: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
3633: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
3634: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
3635: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
3636: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
3637: function CutTime(ADate: TDateTime): { Set time to 00:00:00:00 }
3638: { String to date conversions }
3639: function GetDateFormat(const DateFormat: string): TDateOrder;
3640: function MonthFromName(const S: string; MaxLen: Byte): Byte;
3641: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
3642: function StrToDateFmt(const DateFormat, S: string): TDateTime;
3643: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
3644: function DefDateFormat(FourDigitYear: Boolean): string;
3645: function DefDateMask(BlanksChar: Char; FourDigitYear: Boolean): string;
3646: -----
3647: ***** JvUtils;*****
3648: { GetWordOnPos returns Word from string, S, on the cursor position, P}
3649: function GetWordOnPos(const S: string; const P: Integer): string;
3650: { GetWordOnPosEx work like GetWordOnPos function, also returns Word position in iBeg, iEnd variables }
3651: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3652: { SubStr returns substring from string, S, separated with Separator string}
3653: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3654: { SubStrEnd same to previous function but Index numerated from the end of string }
3655: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3656: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
3657: function SubWord(P: PChar; var P2: PChar): string;
3658: { NumberByWord returns the text representation of
3659:   the number, N, in normal russian language. Was typed from Monitor magazine }
3660: function NumberByWord(const N: Longint): string;
3661: // function CurrencyByWord(Value : Currency) : string; GetLineByPos returns Line number, there
3662: //the symbol Pos is pointed. Lines separated with #13 symbol }

```

```

3663: function GetLineByPos(const S: string; const Pos: Integer): Integer;
3664: { GetXYByPos is same to previous function, but returns X position in line too}
3665: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3666: { ReplaceString searches for all substrings, OldPattern,in a string, S, replaces them with NewPattern }
3667: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3668: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3669: function ConcatSep(const S, S2, Separator: string): string;
3670: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
3671: function ConcatLeftSep(const S, S2, Separator: string): string;
3672: { MinimizeString truncs long string, S, and appends '...' symbols, if Length of S is more than MaxLen }
3673: function MinimizeString(const S: string; const MaxLen: Integer): string;
3674: { Next 4 function for russian chars transliterating.
3675:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes works sucks }
3676: procedure Dos2Win(var S: string);
3677: procedure Win2Dos(var S: string);
3678: function Dos2WinRes(const S: string): string;
3679: function Win2DOSRes(const S: string): string;
3680: function Win2Koi(const S: string): string;
3681: { Spaces returns string consists on N space chars }
3682: function Spaces(const N: Integer): string;
3683: { AddSpaces add spaces to string, S, if it Length is smaller than N }
3684: function AddSpaces(const S: string; const N: Integer): string;
3685: { function LastDate for russian users only } { returns date relative to current date: '' }
3686: function LastDate(const Dat: TDateTime): string;
3687: { CurrencyToStr format currency, Cur, using ffCurrency float format}
3688: function CurrencyToStr(const Cur: currency): string;
3689: { Cmp compares two strings and returns True if they are equal. Case-insensitive.}
3690: function Cmp(const S1, S2: string): Boolean;
3691: { StringCat add S2 string to S1 and returns this string }
3692: function StringCat(var S1: string; S2: string): string;
3693: { HasChar returns True, if Char, Ch, contains in string, S }
3694: function HasChar(const Ch: Char; const S: string): Boolean;
3695: function HasAnyChar(const Chars: string; const S: string): Boolean;
3696: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3697: function CountOfChar(const Ch: Char; const S: string): Integer;
3698: function DefStr(const S: string; Default: string): string;
3699: {**** files routines}
3700: { GetWinDir returns Windows folder name }
3701: function GetWinDir: TFileName;
3702: function GetSysDir: String;
3703: { GetTempDir returns Windows temporary folder name }
3704: function GetTempDir: string;
3705: { GetTempFileName returns temporary file name on drive, there FileName is placed }
3706: function GenTempFileName(FileName: string): string;
3707: { GenTempFileNameExt same to previous function, but returning filename has given extension, FileExt }
3708: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
3709: { ClearDir clears folder Dir }
3710: function ClearDir(const Dir: string): Boolean;
3711: { DeleteDir clears and than delete folder Dir }
3712: function DeleteDir(const Dir: string): Boolean;
3713: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3714: function FileEquMask(FileName, Mask: TFileName): Boolean;
3715: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3716:   Masks must be separated with comma (':') }
3717: function FileEquMasks(FileName, Masks: TFileName): Boolean;
3718: procedure DeleteFiles(const Folder: TFileName; const Masks: string);
3719: { LZFileExpand expand file, FileSource into FileDest.File must be compressed,using MS Compress program }
3720: function LZFileExpand(const FileSource, FileDest: string): Boolean;
3721: { FileGetInfo finds SearchRec record for specified file attributes}
3722: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
3723: { HasSubFolder returns True, if folder APath contains other folders }
3724: function HasSubFolder(APath: TFileName): Boolean;
3725: { IsEmptyFolder returns True, if there are no files or folders in given folder, APPath}
3726: function IsEmptyFolder(APath: TFileName): Boolean;
3727: { AddSlash add slash Char to Dir parameter, if needed }
3728: procedure AddSlash(var Dir: TFileName);
3729: { AddSlash returns string with added slash Char to Dir parameter, if needed }
3730: function AddSlash2(const Dir: TFileName): string;
3731: { AddPath returns FileName with Path, if FileName not contain any path }
3732: function AddPath(const FileName, Path: TFileName): TFileName;
3733: function AddPaths(const PathList, Path: string): string;
3734: function ParentPath(const Path: TFileName): TFileName;
3735: function FindInPath(const FileName, PathList: string): TFileName;
3736: function FindinPaths(const fileName, paths: String): String;
3737: {$IFDEF BCB1}
3738: { BrowseForFolder displays Browse For Folder dialog }
3739: function BrowseForFolder(const Handle: HWND; const Title: string; var Folder: string): Boolean;
3740: {$ENDIF BCB1}
3741: Function BrowseForFolder(const ATitle: string; AllowCreate : Boolean; var ADirectory : string;
AHelpContext : THelpContext) : Boolean;
3742: Function BrowseForComputer(const ATitle : string; AllowCreate : Boolean; var ADirectory : string;
AHelpContext : THelpContext) : Boolean;
3743: Function BrowseDirectory(var AFolderName:string;const DlgText:string;AHelpContext:THelpContext):Boolean;
3744: Function BrowseComputer(var AComputerName:string;const DlgText:string;AHelpContext:THelpContext):Boolean;
3745:
3746: { DeleteReadOnlyFile clears R/O file attribute and delete file }
3747: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
3748: { HasParam returns True, if program running with specified parameter, Param }
3749: function HasParam(const Param: string): Boolean;

```

```

3750: function HasSwitch(const Param: string): Boolean;
3751: function Switch(const Param: string): string;
3752: { ExePath returns ExtractFilePath(ParamStr(0)) }
3753: function ExePath: TFileName;
3754: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
3755: function FileTimeToDateTime(const FT: TFileTime): TDateTime;
3756: function MakeValidFileName(const FileName: TFileName; const ReplaceBadChar: Char): TFileName;
3757: {**** Graphic routines }
3758: { TTFontSelected returns True, if True Type font is selected in specified device context }
3759: function TTFontSelected(const DC: HDC): Boolean;
3760: { TrueInflateRect inflates rect in other method, than InflateRect API function }
3761: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
3762: {**** Windows routines }
3763: { SetWindowTop put window to top without recreating window }
3764: procedure SetWindowTop(const Handle: HWND; const Top: Boolean);
3765: {**** other routines }
3766: { KeyPressed returns True, if Key VK is now pressed }
3767: function KeyPressed(VK: Integer): Boolean;
3768: procedure SwapInt(var Int1, Int2: Integer);
3769: function IntPower(Base, Exponent: Integer): Integer;
3770: function ChangeTopException(E: TObject): TObject;
3771: function StrToBool(const S: string): Boolean;
3772: {$IFDEF COMPILER3_UP}
3773: { AnsiStrLIComp compares S1 to S2, without case-sensitivity, up to a maximum
3774:   Length of MaxLen bytes. The compare operation is controlled by the
3775:   current Windows locale. The return value is the same as for CompareStr. }
3776: function AnsiStrLIComp(S1, S2: PChar; MaxLen: Cardinal): Integer;
3777: function AnsiStrICmp(S1, S2: PChar): Integer;
3778: {$ENDIF}
3779: function Var2Type(V: Variant; const VarType: Integer): Variant;
3780: function VarToInt(V: Variant): Integer;
3781: function VarToFloat(V: Variant): Double;
3782: { following functions are not documented because they are don't work properly , so don't use them }
3783: function ReplaceSokr1(S: string; const Word, Frase: string): string;
3784: { ReplaceSokr1 is full equal to ReplaceString function - only for compatibility - don't use }
3785: { GetSubStr is full equal to SubStr function - only for compatibility - don't use }
3786: function GetSubStr(const S: string; const Index: Integer; const Separator: Char): string;
3787: function GetParameter: string;
3788: function GetLongFileName(FileName: string): string;
3789: {* from FileCtrl}
3790: function DirectoryExists(const Name: string): Boolean;
3791: procedure ForceDirectories(Dir: string);
3792: {# from FileCtrl}
3793: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
3794: function GetComputerID: string;
3795: function GetComputerName: string;
3796: {**** string routines }
3797: { ReplaceAllSokr searches for all substrings, Words,in a string, S, and replaces them with Frases with the
3798:   same Index.Also see RAUtilsW.ReplaceSokr1 function }
3799: function ReplaceAllSokr(S: string; Words, Frases: TStrings): string;
3800: { ReplaceSokr searches the Word in a string, S, on PosBeg position,
3801:   in the list, Words, and if founds, replaces this Word with string from another list, Frases, with the
3802:   same Index, and then update NewSelStart variable }
3803: function ReplaceSokr(S:string;PosBeg,Len:Integer;Words,Frases:TStrings;var NewSelStart:Integer): string;
3804: { CountOfLines calculates the lines count in a string,each line separated from another with CrLf sequence }
3805: function CountOfLines(const S: string): Integer;
3806: { DeleteEmptyLines deletes all empty lines from strings, Ss. Lines contained only spaces also deletes. }
3807: procedure DeleteEmptyLines(Ss: TStrings);
3808: { SQLAddWhere addes or modifies existing where-statement, where, to the strings, SQL.
3809:   Note: If strings SQL already contains where-statement, it must be started on begining of any line }
3810: procedure SQLAddWhere(SQL: TStrings; const Where: string);
3811: { ResSaveToFile save resource named as Name with Typ type into file FileName.
3812:   Resource can be compressed using MS Compress program}
3813: function ResSaveToFile(const Typ,Name: string; const Compressed:Boolean; const FileName: string): Boolean;
3814: function ResSaveToFileEx(Inst:HINST;Typ,Name:PChar;const Compressed:Bool;const FileName:string): Bool;
3815: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
3816: function ExecuteJ(const CommandLine, WorkingDirectory: string): Integer;
3817: { IniReadSection read section, Section, from ini-file,
3818:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
3819:   Note: TIniFile.ReadSection function reads only strings with '=' symbol.}
3820: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
3821: { LoadTextFile load text file, FileName, into string }
3822: function LoadTextFile(const FileName: TFileName): string;
3823: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3824: { ReadFolder reads files list from disk folder, Folder, that are equal Mask, into strings, FileList}
3825: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
3826: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
3827: {$IFDEF COMPILER3_UP}
3828: { TargetFileName - if FileName is ShortCut returns filename ShortCut linked to }
3829: function TargetFileName(const FileName: TFileName): TFileName;
3830: { return filename ShortCut linked to }
3831: function ResolveLink(const hWnd: HWND; const LinkFile: TFileName; var FileName: TFileName): HRESULT;
3832: {$ENDIF COMPILER3_UP}
3833: {**** Graphic routines - }
3834: { LoadIcoToImage loads two icons from resource named NameRes,into two image lists ALarge and ASmall}
3835: procedure LoadIcoToImage(ALarge, ASmall: TImageList; const NameRes: string);
3836: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }

```

```

3837: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
3838: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
3839: function RATextOutEx(Canvas:TCanvas; const R,RClip:TRect;const S: string;const CalcHeight:Boolean):Integer;
3840: { RATextCalcHeight calculate needed height to correct output, using RATextOut or RATextOutEx functions }
3841: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
3842: { Cinema draws some visual effect }
3843: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
3844: { Roughed fills rect with special 3D pattern }
3845: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
3846: { 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 }
3847: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
3848: { TextWidth calculate text for writing using standard desktop font }
3849: function TextWidth(AString: string): Integer;
3850: { DefineCursor load cursor from resource, and return available cursor number, assigned to it }
3851: function DefineCursor(Identifier: PChar): TCursor;
3852: {**** other routines - }
3853: { FindFormByClass returns first form specified class, FormClass, owned by Application global variable }
3854: function FindFormByClass(FormClass: TFormClass): TForm;
3855: function FindFormByClassName(FormClassName: string): TForm;
3856: { FindByTag returns the control with specified class, ComponentClass, from WinControl.Controls property,
  having Tag property value, equaled to Tag parameter }
3857: function FindByTag(WinControl:TWinControl;ComponentClass:TComponentClass;const Tag:Integer):TComponent;
3858: { ControlAtPos2 equal to TWInControl.ControlAtPos function, but works better }
3860: function ControlAtPos2(Parent: TWinControl; X, Y: Integer): TControl;
3861: { RBTAG searches WinControl.Controls for checked RadioButton and returns its Tag property value }
3862: function RBTAG(Parent: TWinControl): Integer;
3863: { AppMinimized returns True, if Application is minimized }
3864: function AppMinimized: Boolean;
3865: { MessageBox is Application.MessageBox with string (not PChar) parameters.
  if Caption parameter = '', it replaced with Application.Title }
3867: function MessageBoxJ(const Msg: string; Caption: string;const Flags: Integer): Integer;
3868: function MsgDlg2(const Msg, ACaption: string; DlgType: TMsgDlgType;
  Buttons: TMsgDlgButtons; HelpContext: Integer; Control: TWinControl): Integer;
3870: function MsgDlgDef(const Msg, ACaption: string; DlgType: TMsgDlgType;
  Buttons: TMsgDlgButtons; DefButton:TMsgDlgBtn; HelpContext: Integer;Control: TWinControl): Integer;
3872: { Delay stop program execution to MSec msec }
3873: procedure Delay(MSec: Longword);
3874: procedure CenterHor(Parent: TControl; MinLeft: Integer; Controls: array of TControl);
3875: procedure EnableControls(Control: TWinControl; const Enable: Boolean);
3876: procedure EnableMenuItem(MenuItem: TMenuItem; const Tag: Integer; const Enable: Boolean);
3877: procedure ExpandWidth(Parent: TControl; MinWidth: Integer; Controls: array of TControl);
3878: function PanelBorder(Panel: TCustomPanel): Integer;
3879: function Pixels(Control: TControl; APixels: Integer): Integer;
3880: procedure SetChildPropOrd(Owner: TComponent; PropName: string; Value: Longint);
3881: procedure Error(const Msg: string);
3882: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
  const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
3884: {ex. Text parameter:'Item 1 <b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green <c:blue>blue</i>'}
3885: function ItemHtDraw(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
  const HideSelColor: Boolean): string;
3887: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
  const HideSelColor: Boolean): Integer;
3889: function ItemHtPlain(const Text: string): string;
3890: { ClearList - clears list of TObject }
3891: procedure ClearList(List: TList);
3892: procedure MemStreamToClipboard(MemStream: TMemoryStream; const Format: Word);
3893: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
3894: { RTTI support }
3895: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
3896: function GetPropStr(Obj: TObject; const PropName: string): string;
3897: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
3898: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
3899: procedure PrepareIniSection(SS: TStrings);
3900: { following functions are not documented because they are don't work properly, so don't use them }
3901: {$IFDEF COMPILER2}
3902: function CompareMem(P1, P2: Pointer; Length: Integer): Boolean; assembler;
3903: {$ENDIF}
3904:
3905: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
3906: begin
3907: Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)
3908: Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)
3909: Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
  Accept:Bool;Sorted:Bool;
3910: Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)
3911: Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)
3912: Procedure BoxSetItem( List : TWinControl; Index : Integer)
3913: Function BoxGetFirstSelection( List : TWinControl) : Integer
3914: Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean
3915: end;
3916:
3917: procedure SIRegister_JvCsvParse(CL: TPSPascalCompiler);
3918: begin
3919: Const ('MaxInitStrNum','LongInt' ( 9));
3920: Function JvAnsiStrSplit( const InString : AnsiString; const SplitChar, QuoteChar: AnsiChar; var OutStrings
  : array of AnsiString; MaxSplit : Integer) : Integer
3921: Function JvStrSplit( const InString : string; const SplitChar, QuoteChar : Char; var OutStrings : array of
  string; MaxSplit : Integer) : Integer

```

```

3922: Function JvAnsiStrSplitStrings(const InStr: AnsiString; const SplitChar,
3923:   QuoteChar: AnsiChar; OutStrs: TStrings): Integer;
3924: Function JvAnsiStrStrip( S : AnsiString ) : AnsiString;
3925: Function JvStrStrip( S : string ) : string;
3926: Function GetString( var Source : AnsiString; const Separator : AnsiString ) : AnsiString;
3927: Function PadString( const S : AnsiString; Len : Integer; PadChar : AnsiChar ) : AnsiString;
3928: Function StrEatWhiteSpace( const S : string ) : string;
3929: Function HexToAscii( const S : AnsiString ) : AnsiString;
3930: Function AsciiToHex( const S : AnsiString ) : AnsiString;
3931: Function StripQuotes( const S1 : AnsiString ) : AnsiString;
3932: Function ValidNumericLiteral( S1 : PAnsiChar ) : Boolean;
3933: Function ValidIntLiteral( S1 : PAnsiChar ) : Boolean;
3934: Function ValidHexLiteral( S1 : PAnsiChar ) : Boolean;
3935: Function HexPCharToInt( S1 : PAnsiChar ) : Integer;
3936: Function ValidStringLiteral( S1 : PAnsiChar ) : Boolean;
3937: Function StripPCharQuotes( S1 : PAnsiChar ) : AnsiString;
3938: Function JvValidIdentifierAnsi( S1 : PAnsiChar ) : Boolean;
3939: Function JvValidIdentifier( S1 : String ) : Boolean;
3940: Function JvEndChar( X : AnsiChar ) : Boolean;
3941: Procedure JvGetToken( S1, S2 : PAnsiChar );
3942: Function IsExpressionKeyword( S1 : PAnsiChar ) : Boolean;
3943: Function IsKeyword( S1 : PAnsiChar ) : Boolean;
3944: Function JvValidVarReference( S1 : PAnsiChar ) : Boolean;
3945: Function GetParenthesis( S1, S2 : PAnsiChar ) : Boolean;
3946: Procedure JvGetVarReference( S1, S2, SIdx : PAnsiChar );
3947: Procedure JvEatWhitespaceChars( S1 : PAnsiChar );
3948: Procedure JvEatWhitespaceChars1( S1 : PWideChar );
3949: Function GetTokenCount : Integer;
3950: Procedure ResetTokenCount;
3951: end;
3952;
3953: procedure SIRegister_JvDBQueryParamsForm(CL: TPPSPascalCompiler);
3954: begin
3955:   SIRegister_TJvQueryParamsDialog(CL);
3956:   Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext ) : Boolean;
3957: end;
3958;
3959: ***** JvStringUtil / JvStringUtilities *****
3960: function FindNotBlankCharPos(const S: string): Integer;
3961: function AnsiChangeCase(const S: string): string;
3962: function GetWordOnPos(const S: string; const P: Integer): string;
3963: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3964: function Cmp(const S1, S2: string): Boolean;
3965: { Spaces returns string consists on N space chars }
3966: function Spaces(const N: Integer): string;
3967: { HasChar returns True, if char, Ch, contains in string, S }
3968: function HasChar(const Ch: Char; const S: string): Boolean;
3969: function HasAnyChar(const Chars: string; const S: string): Boolean;
3970: { SubStr returns substring from string, S, separated with Separator string}
3971: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3972: { SubStrEnd same to previous function but Index numerated from the end of string }
3973: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3974: { ReplaceString searches for all substrings, OldPattern, in a string, S, replaces them with NewPattern }
3975: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3976: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3977: { GetXYByPos is same to previous function, but returns X position in line too}
3978: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3979: { AddSlash returns string with added slash char to Dir parameter, if needed }
3980: function AddSlash2(const Dir: TFileName): string;
3981: { AddPath returns FileName with Path, if FileName not contain any path }
3982: function AddPath(const FileName, Path: TFileName): TFileName;
3983: { ExePath returns ExtractFilePath(ParamStr(0)) }
3984: function ExePath: TFileName;
3985: function LoadTextFile(const FileName: TFileName): string;
3986: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3987: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3988: function ConcatSep(const S, S2, Separator: string): string;
3989: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3990: function FileEquMask(FileName, Mask: TFileName): Boolean;
3991: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3992:   Masks must be separated with comma (';') }
3993: function FileEquMasks(FileName, Masks: TFileName): Boolean;
3994: function StringEndsWith(const Str, SubStr: string): Boolean;
3995: function ExtractFilePath2(const FileName: string): string;
3996: function StrToOem(const AnsiStr: string): string;
3997: { StrToOem translates a string from the Windows character set into the OEM character set. }
3998: function OemToAnsiStr(const OemStr: string): string;
3999: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4000: function IsEmptyStr(const S: string; const EmptyChars: TCharSet): Boolean;
4001: { EmptyStr returns true if the given string contains only character from the EmptyChars. }
4002: function ReplaceStr(const S, Srch, Replace: string): string;
4003: { Returns string with every occurrence of Srch string replaced with Replace string. }
4004: function DelSpace(const S: string): string;
4005: { DelSpace return a string with all white spaces removed. }
4006: function DelChars(const S: string; Chr: Char): string;
4007: { DelChars return a string with all Chr characters removed. }
4008: function DelBSpace(const S: string): string;
4009: { DelBSpace trims leading spaces from the given string. }

```

```

4010: function DelESpace(const S: string): string;
4011: { DelESpace trims trailing spaces from the given string. }
4012: function DelRSpace(const S: string): string;
4013: { DelRSpace trims leading and trailing spaces from the given string. }
4014: function DelSpace1(const S: string): string;
4015: { DelSpace1 return a string with all non-single white spaces removed. }
4016: function Tab2Space(const S: string; Numb: Byte): string;
4017: { Tab2Space converts any tabulation character in the given string to the Numb spaces characters. }
4018: function NPos(const C: string; S: string; N: Integer): Integer;
4019: { NPos searches for a N-th position of substring C in a given string. }
4020: function MakeStr(C: Char; N: Integer): string;
4021: function MS(C: Char; N: Integer): string;
4022: { MakeStr return a string of length N filled with character C. }
4023: function AddChar(C: Char; const S: string; N: Integer): string;
4024: { AddChar return a string left-padded to length N with characters c. }
4025: function AddCharR(C: Char; const S: string; N: Integer): string;
4026: { AddCharR return a string right-padded to length N with characters C. }
4027: function LeftStr(const S: string; N: Integer): string;
4028: { LeftStr return a string right-padded to length N with blanks. }
4029: function RightStr(const S: string; N: Integer): string;
4030: { RightStr return a string left-padded to length N with blanks. }
4031: function CenterStr(const S: string; Len: Integer): string;
4032: { CenterStr centers the characters in the string based upon the Len specified. }
4033: function CompStr(const S1, S2: string): Integer;
4034: { CompStr compares S1 to S2, case-sensitivity. return val is -1 if S1 < S2,0 if S1 = S2,or 1 if S1>S2. }
4035: function CompText(const S1, S2: string): Integer;
4036: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4037: function Copy2Symb(const S: string; Symb: Char): string;
4038: { Copy2Symb returns a substring of a string S from beginning to first character Symb. }
4039: function Copy2SymbDel(var S: string; Symb: Char): string;
4040: { Copy2SymbDel returns a substr of string S from to first character Symb and removes substring from S. }
4041: function Copy2Space(const S: string): string;
4042: { Copy2Space returns a substring of a string S from beginning to first white space. }
4043: function Copy2SpaceDel(var S: string): string;
4044: { Copy2SpaceDel returns a substring of a string S from begining to first
4045:   white space and removes this substring from S. }
4046: function AnsiProperCase(const S: string; const WordDelims: TCharSet): string;
4047: { Returns string, with the first letter of each word in uppercase,
4048:   all other letters in lowercase. Words are delimited by WordDelims. }
4049: function WordCount(const S: string; const WordDelims: TCharSet): Integer;
4050: { WordCount given a set of word delimiters, returns number of words in S. }
4051: function WordPosition(const N: Integer; const S: string; const WordDelims: TCharSet): Integer;
4052: { Given a set of word delimiters, returns start position of N'th word in S. }
4053: function ExtractWord(N: Integer; const S: string; const WordDelims: TCharSet): string;
4054: function ExtractWordPos(N: Integer; const S: string; const WordDelims: TCharSet; var Pos: Integer): string;
4055: function ExtractDelimited(N: Integer; const S: string; const Delims: TCharSet): string;
4056: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4057:   delimiters, return the N'th word in S. }
4058: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TCharSet): string;
4059: { ExtractSubstr given set of word delimiters, returns the substring from S,that started from position Pos.}
4060: function IsWordPresent(const W, S: string; const WordDelims: TCharSet): Boolean;
4061: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4062: function QuotedString(const S: string; Quote: Char): string;
4063: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4064: function ExtractQuotedString(const S: string; Quote: Char): string;
4065: { ExtractQuotedString removes the Quote characters from the beginning and end of a quoted string,
4066:   and reduces pairs of Quote characters within the quoted string to a single character. }
4067: function FindPart(const HelpWilds, InputStr: string): Integer;
4068: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4069: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4070: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4071: function XorString(const Key, Src: ShortString): ShortString;
4072: function XorEncode(const Key, Source: string): string;
4073: function XorDecode(const Key, Source: string): string;
4074: { ** Command line routines ** }
4075: {$IFNDEF COMPILER4_UP}
4076: function FindCmdLineSwitch(const Switch: string; SwitchChars: TCharSet; IgnoreCase: Boolean): Boolean;
4077: {$ENDIF}
4078: function GetCmdLineArg(const Switch: string; SwitchChars: TCharSet): string;
4079: { ** Numeric string handling routines ** }
4080: function Numb2USA(const S: string): string;
4081: { Numb2USA converts numeric string S to USA-format. }
4082: function Dec2Hex(N: Longint; A: Byte): string;
4083: function D2H(N: Longint; A: Byte): string;
4084: { Dec2Hex converts the given value to a hexadecimal string representation
4085:   with the minimum number of digits (A) specified. }
4086: function Hex2Dec(const S: string): Longint;
4087: function H2D(const S: string): Longint;
4088: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4089: function Dec2Numb(N: Longint; A, B: Byte): string;
4090: { Dec2Numb converts the given value to a string representation with the
4091:   base equal to B and with the minimum number of digits (A) specified. }
4092: function Numb2Dec(S: string; B: Byte): Longint;
4093: { Numb2Dec converts the given B-based numeric string to the corresponding integer value. }
4094: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4095: { IntToBin converts given value to a bin string representation with min number of digits specified. }
4096: function IntToRoman(Value: Longint): string;
4097: { IntToRoman converts the given value to a roman numeric string representation. }
4098: function RomanToInt(const S: string): Longint;

```

```

4099: { RomanToInt converts the given string to an integer value. If the string
4100:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4101: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
4102: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
4103: ***** JvFileUtil;*****
4104: procedure CopyFileJ(const FileName, DestName: string; ProgressControl: TControl);
4105: procedure CopyFileEx(const FileName, DestName:string;OverwriteReadOnly,ShellDialog:Boolean;ProgressControl: TControl);
4106: procedure MoveFile(const FileName, DestName: TFileName);
4107: procedure MoveFileEx(const FileName, DestName: TFileName; ShellDialog: Boolean);
4108: {$IFDEF COMPILER4_UP}
4109: function GetFileSize(const FileName: string): Int64;
4110: {$ELSE}
4111: function GetFileSize(const FileName: string): Longint;
4112: {$ENDIF}
4113: function FileDateTime(const FileName: string): TDateTime;
4114: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4115: function DeleteFiles(const FileMask: string): Boolean;
4116: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4117: function ClearDir(const Path: string; Delete: Boolean): Boolean;
4118: function NormalDir(const DirName: string): string;
4119: function RemoveBackSlash(const DirName: string): string;
4120: function ValidFileName(const FileName: string): Boolean;
4121: function DirExists(Name: string): Boolean;
4122: procedure ForceDirectories(Dir: string);
4123: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer;
4124: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4125: {$IFDEF COMPILER4_UP}
4126: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4127: {$ENDIF}
4128: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer;
4129: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4130: {$IFDEF COMPILER4_UP}
4131: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4132: {$ENDIF}
4133: function GetTempDir: string;
4134: function GetWindowsDir: string;
4135: function GetSystemDir: string;
4136: function BrowseDirectory(var AFolderName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4137: {$IFDEF WIN32}
4138: function BrowseComputer(var ComputerName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4139: function ShortToLongFileName(const ShortName: string): string;
4140: function ShortToLongPath(const ShortName: string): string;
4141: function LongToShortFileName(const LongName: string): string;
4142: function LongToShortPath(const LongName: string): string;
4143: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4144: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4145: {$ENDIF WIN32}
4146: {$IFDEF COMPILER3_UP}
4147: function IsPathDelimiter(const S: string; Index: Integer): Boolean;
4148: {$ENDIF}
4149: Function CreateCalculatorForm( AOwner : TComponent; AHelpContext : THelpContext ) : TJvCalculatorForm;
4150: Function CreatePopupCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode ) : TWinControl;
4151: Function CreatePopupCalculator( AOwner : TComponent ) : TWinControl;
4152: Procedure SetupPopupCalculator( PopupCalc : TWinControl; APrecision : Byte; ABeepOnError : Boolean );
4153: //*****procedure SIRegister_VarHlpr(CL: TPSPPascalCompiler);
4154: Procedure VariantClear( var V : Variant );
4155: Procedure VariantArrayRedim( var V : Variant; High : Integer );
4156: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer );
4157: Procedure VariantCpy( const src : Variant; var dst : Variant );
4158: Procedure VariantAdd( const src : Variant; var dst : Variant );
4159: Procedure VariantSub( const src : Variant; var dst : Variant );
4160: Procedure VariantMul( const src : Variant; var dst : Variant );
4161: Procedure VariantDiv( const src : Variant; var dst : Variant );
4162: Procedure VariantMod( const src : Variant; var dst : Variant );
4163: Procedure VariantAnd( const src : Variant; var dst : Variant );
4164: Procedure VariantOr( const src : Variant; var dst : Variant );
4165: Procedure VariantXor( const src : Variant; var dst : Variant );
4166: Procedure VariantShl( const src : Variant; var dst : Variant );
4167: Procedure VariantShr( const src : Variant; var dst : Variant );
4168: Function VariantAdd2( const V1 : Variant; const V2 : Variant ) : Variant;
4169: Function VariantSub2( const V1 : Variant; const V2 : Variant ) : Variant;
4170: Function VariantMul2( const V1 : Variant; const V2 : Variant ) : Variant;
4171: Function VariantDiv2( const V1 : Variant; const V2 : Variant ) : Variant;
4172: Function VariantMod2( const V1 : Variant; const V2 : Variant ) : Variant;
4173: Function VariantAnd2( const V1 : Variant; const V2 : Variant ) : Variant;
4174: Function VariantOr2( const V1 : Variant; const V2 : Variant ) : Variant;
4175: Function VariantXor2( const V1 : Variant; const V2 : Variant ) : Variant;
4176: Function VariantShl2( const V1 : Variant; const V2 : Variant ) : Variant;
4177: Function VariantShr2( const V1 : Variant; const V2 : Variant ) : Variant;
4178: Function VariantNot( const V1 : Variant ) : Variant;
4179: Function VariantNeg( const V1 : Variant ) : Variant;
4180: Function VariantGetElement( const V : Variant; il : integer ) : Variant;
4181: Function VariantGetElement1( const V : Variant; il, i2 : integer ) : Variant;
4182: Function VariantGetElement2( const V : Variant; il, i2, i3 : integer ) : Variant;
4183: Function VariantGetElement3( const V : Variant; il, i2, i3, i4 : integer ) : Variant;
4184: Function VariantGetElement4( const V : Variant; il, i2, i3, i4, i5 : integer ) : Variant;
4185: Procedure VariantPutElement( var V : Variant; const data : Variant; il : integer );

```

```

4187: Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
4188: Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
4189: Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
4190: Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
4191: end;
4192:
4193: *****unit uPSI_JvgUtils;*****
4194: function IsEven(I: Integer): Boolean;
4195: function InchesToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4196: function CentimetersToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4197: procedure SwapInt(var I1, I2: Integer);
4198: function Spaces(Count: Integer): string;
4199: function DupStr(const Str: string; Count: Integer): string;
4200: function DupChar(C: Char; Count: Integer): string;
4201: procedure Msg(const AMsg: string);
4202: function RectW(R: TRect): Integer;
4203: function RectH(R: TRect): Integer;
4204: function IncColor(AColor: Longint; AOffset: Byte): Longint;
4205: function DecColor(AColor: Longint; AOffset: Byte): Longint;
4206: function IsItAFilledBitmap(Bmp: TBitmap): Boolean;
4207: procedure DrawTextInRectWithAlign(DC: HDC; R: TRect; const Text: string;
4208:   HAlign: TglHorAlign; VAlign: TglVertAlign; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
4209: procedure DrawTextInRect(DC: HDC; R: TRect; const Text:string;Style:TglTextStyle;Fnt:TFont;Flags: UINT);
4210: procedure ExtTextOutExt(DC: HDC; X, Y: Integer; R: TRect; const Text: string;
4211:   Style: TglTextStyle; ADelineted, ASupress3D: Boolean; FontColor, DelinColor, HighlightColor, ShadowColor:
4212:   TColor; Illumination: TJvgIllumination; Gradient: TJvgGradient; Font: TFont);
4213: procedure DrawBox(DC:HDC; var R:TRect; Style:TglBoxStyle;BackgrColor: Longint; ATransparent: Boolean);
4214: function DrawBoxEx(DC: HDC; ARect: TRect; Borders: TglSides;
4215:   BevelInner, BevelOuter: TPanelBevel; Bold:Boolean; BackgrColor: Longint; ATransparent: Boolean): TRect;
4216: procedure GradientBox(DC: HDC; R: TRect; Gradient: TJvgGradient; PenStyle, PenWidth: Integer);
4217: procedure ChangeBitmapColor(Bitmap: TBitmap; FromColor, ToColor: TColor);
4218: procedure DrawBitmapExt(DC: HDC; { DC - background & result}
4219:   Sourcebitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
4220:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4221:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4222: procedure CreateBitmapExt(DC: HDC; { DC - background & result}
4223:   Sourcebitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
4224:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4225:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4226: function GetParentForm(Control: TControl): TForm;
4227: procedure GetWindowImageFrom(Control:TWinControl;X,Y:Integer;ADrawSelf,ADrawChildWindows:Boolean;DC:HDC);
4228: procedure GetWindowImage(Control: TWinControl; ADrawSelf, ADrawChildWindows: Boolean; DC: HDC);
4229: procedure GetParentImageRect(Control: TControl; Rect: TRect; DC: HDC);
4230: function CreateRotatedFont(F: TFont; Escapement: Integer): HFONT;
4231: function FindMainWindow(const AWndClass, AWndTitle: string): THandle;
4232: procedure CalcShadowAndHighlightColors(BaseColor: TColor; Colors: TJvgLabelColors);
4233: function CalcMathString(AExpression: string): Single;
4234: function IIF(AExpression: Boolean; IfTrue, IfFalse: Variant): Variant; overload;
4235: function IIF(AExpression: Boolean; const IfTrue, IfFalse: string): string; overload;
4236: function GetTransparentColor(Bitmap: TBitmap; AutoTrColor: TglAutoTransparentColor): TColor;
4237: procedure TypeStringOnKeyboard(const S: string);
4238: function NextStringGridCell(Grid: TStringGrid): Boolean;
4239: procedure DrawTextExtAligned(Canvas: TCanvas; const
4240:   Text:string;R:TRect;Alignment:TglAlignment;WordWrap:Boolean);
4241: procedure LoadComponentFromTextFile(Component: TComponent; const FileName: string);
4242: function SaveComponentToTextFile(Component: TComponent; const FileName: string);
4243: function ComponentToString(Component: TComponent): string;
4244: function StringToComponent(Component: TComponent; const Value: string);
4245: function PlayWaveResource(const ResName: string): Boolean;
4246: function UserName: string;
4247: function ComputerName: string;
4248: function CreateIniFileName: string;
4249: function ExpandString(const Str: string; Len: Integer): string;
4250: function Transliterate(const Str: string; RusToLat: Boolean): string;
4251: function IsSmallFonts: Boolean;
4252: function SystemColorDepth: Integer;
4253: function GetFileTypeJ(const FileName: string): TglFileType;
4254: function GetFileType(hFile : THandle) : DWORD';
4255: function FindControlAtPt(Control: TWinControl; Pt: TPoint; MinClass: TClass): TControl;
4256: function StrPosExt(const Str1, Str2: PChar; Str2Len: DWORD): PChar;
4257: { ****Utility routines of unit classes}
4258: function LineStart(Buffer, BufPos: PChar): PChar;
4259: function ExtractStrings(Separators,WhiteSpace: TSysCharSet; Content: PChar;'+'
4260:   'Strings: TStrings): Integer
4261: function TestStreamFormat(Stream : TStream) : TStreamOriginalFormat;
4262: procedure RegisterClass(AClass : TPersistentClass);
4263: procedure RegisterClasses(AClasses : array of TPersistentClass);
4264: procedure RegisterClassAlias(AClass : TPersistentClass; const Alias : string);
4265: procedure UnRegisterClass(AClass : TPersistentClass);
4266: procedure UnRegisterClasses(AClasses : array of TPersistentClass);
4267: procedure UnRegisterModuleClasses(Module : HMODULE);
4268: function FindGlobalComponent(const Name : string) : TComponent;
4269: function IsUniqueGlobalComponentName(const Name : string) : Boolean;
4270: function InitInheritedComponent(Instance : TComponent; RootAncestor : TClass) : Boolean;
4271: function InitComponentRes(const ResName : string; Instance : TComponent) : Boolean;
4272: function ReadComponentRes(const ResName : string; Instance : TComponent) : TComponent;
4273: function ReadComponentResEx(HInstance : THandle; const ResName : string) : TComponent;

```

```

4274: Function ReadComponentResFile( const FileName : string; Instance : TComponent ) : TComponent
4275: Procedure WriteComponentResFile( const FileName : string; Instance : TComponent )
4276: Procedure GlobalFixupReferences
4277: Procedure GetFixupReferenceNames( Root : TComponent; Names : TStrings )
4278: Procedure GetFixupInstanceNames(Root : TComponent; const ReferenceRootName string; Names : TStrings)
4279: Procedure RedirectFixupReferences( Root : TComponent; const OldRootName, NewRootName : string )
4280: Procedure RemoveFixupReferences( Root : TComponent; const RootName : string )
4281: Procedure RemoveFixups( Instance : TPersistent )
4282: Function FindNestedComponent( Root : TComponent; const NamePath : string ) : TComponent
4283: Procedure BeginGlobalLoading
4284: Procedure NotifyGlobalLoading
4285: Procedure EndGlobalLoading
4286: Function GetUltimateOwner1( ACollection : TCollection ) : TPersistent;
4287: Function GetUltimateOwner( APersistent : TPersistent ) : TPersistent;
4288: // AddTypeS('TWndMethod', 'Procedure (var Message : TMessage)'
4289: //Function MakeObjectInstance( Method : TWndMethod ) : Pointer
4290: Procedure FreeObjectInstance( ObjectInstance : Pointer )
4291: // Function AllocateHWnd( Method : TWndMethod ) : HWnd
4292: Procedure DeallocateHWnd( Wnd : HWnd )
4293: Function AncestorIsValid( Ancestor : TPersistent; Root, RootAncestor : TComponent ) : Boolean
4294: *****unit uPSI_SqlTimSt and DB;*****
4295: Procedure VarSQLTimeStampCreate4( var aDest : Variant; const ASQLTimeStamp : TSQLOTimeStamp );
4296: Function VarSQLTimeStampCreate3: Variant;
4297: Function VarSQLTimeStampCreate2( const aValue : string ) : Variant;
4298: Function VarSQLTimeStampCreate1( const aValue : TDateTime ) : Variant;
4299: Function VarSQLTimeStampCreate( const ASQLTimeStamp : TSQLOTimeStamp ) : Variant;
4300: Function VarSQLTimeStamp : TVarType
4301: Function VarIsSQLTimeStamp( const aValue : Variant ) : Boolean;
4302: Function LocalToUTC( var TZInfo : TTTimeZone; var Value : TSQLOTimeStamp ) : TSQLOTimeStamp //beta
4303: Function UTCToLocal( var TZInfo : TTTimeZone; var Value : TSQLOTimeStamp ) : TSQLOTimeStamp //beta
4304: Function VarToSQLTimeStamp( const aValue : Variant ) : TSQLOTimeStamp
4305: Function SQLTimeStampToStr( const Format : string; DateTime : TSQLOTimeStamp ) : string
4306: Function SQLDayOfWeek( const DateTime : TSQLOTimeStamp ) : integer
4307: Function DateTimeToSQLTimeStamp( const DateTime : TDateTime ) : TSQLOTimeStamp
4308: Function SQLTimeStampToDate( const Date : TSQLOTimeStamp ) : TDate
4309: Function TryStrToSQLTimeStamp( const S : string; var TimeStamp : TSQLOTimeStamp ) : Boolean
4310: Function StrToSQLTimeStamp( const S : string ) : TSQLOTimeStamp
4311: Procedure CheckSQLTimeStamp( const ASQLTimeStamp : TSQLOTimeStamp )
4312: Function ExtractFieldName( const Fields : string; var Pos : Integer ) : string;
4313: Function ExtractFieldName( const Fields : WideString; var Pos : Integer ) : WideString;
4314: //Procedure RegisterFields( const FieldClasses : array of TFieldClass )
4315: Procedure DatabaseError( const Message : WideString; Component : TComponent )
4316: Procedure DatabaseErrorFmt( const Message:WideString; const Args:array of const;Component:TComponent )
4317: Procedure DisposeMem( var Buffer, Size : Integer )
4318: Function BuffersEqual( Buf1, Buf2 : Pointer; Size : Integer ) : Boolean
4319: Function GetFieldProperty(DataSet:TDataSet; Control:TComponent; const FieldName: WideString): TField
4320: Function VarTypeToDataType( VarType : Integer ) : TFieldType
4321: *****unit JvStrings;*****
4322: {template functions}
4323: function ReplaceFirst(const SourceStr, FindStr, ReplaceStr: string): string;
4324: function ReplaceLast(const SourceStr, FindStr, ReplaceStr: string): string;
4325: function InsertLastBlock(var SourceStr: string; BlockStr: string): Boolean;
4326: function RemoveMasterBlocks(const SourceStr: string): string;
4327: function RemoveFields(const SourceStr: string): string;
4328: {http functions}
4329: function URL Encode(const Value: AnsiString): AnsiString; // Converts string To A URLEncoded string
4330: function URL Decode(const Value: AnsiString): AnsiString; // Converts string From A URLEncoded string
4331: {set functions}
4332: procedure SplitSet(AText: string; AList: TStringList);
4333: function JoinSet(Alist: TStringList): string;
4334: function FirstOfSet(const AText: string): string;
4335: function LastOfSet(const AText: string): string;
4336: function CountOfSet(const AText: string): Integer;
4337: function SetRotateRight(const AText: string): string;
4338: function SetRotateLeft(const AText: string): string;
4339: function SetPick(const AText: string; AIIndex: Integer): string;
4340: function SetSort(const AText: string): string;
4341: function SetUnion(const Set1, Set2: string): string;
4342: function SetIntersect(const Set1, Set2: string): string;
4343: function SetExclude(const Set1, Set2: string): string;
4344: {replace any <, > etc by &lt; ; &gt;}
4345: function XMLSafe(const AText: string): string;
4346: {simple hash, Result can be used in Encrypt}
4347: function Hash(const AText: string): Integer;
4348: { Base64 encode and decode a string }
4349: function B64Encode(const S: AnsiString): AnsiString;
4350: function B64Decode(const S: AnsiString): AnsiString;
4351: {Basic encryption from a Borland Example}
4352: function Encrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4353: function Decrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4354: {Using Encrypt and Decrypt in combination with B64Encode and B64Decode}
4355: function EncryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4356: function DecryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4357: procedure CSVToTags(Src, Dst: TStringList);
4358: // converts a csv list to a tagged string list
4359: procedure TagsToCSV(Src, Dst: TStringList);
4360: // converts a tagged string list to a csv list
4361: // only fieldnames from the first record are scanned in the other records
4362: procedure ListSelect(Src, Dst: TStringList; const AKey, AValue: string);

```

```

4363: {selects akey=avalue from Src and returns recordset in Dst}
4364: procedure ListFilter(Src: TStringList; const AKey, AValue: string);
4365: {filters Src for akey=avalue}
4366: procedure ListOrderBy(Src: TStringList; const AKey: string; Numeric: Boolean);
4367: {orders a tagged Src list by akey}
4368: function PosStr(const FindString, SourceString: string;
4369:   StartPos: Integer = 1): Integer;
4370: { PosStr searches the first occurrence of a substring FindString in a string
4371:   given by SourceString with case sensitivity (upper and lower case characters
4372:   are differed). This function returns the index value of the first character
4373:   of a specified substring from which it occurs in a given string starting with
4374:   StartPos character index. If a specified substring is not found Q_PosStr
4375:   returns zero. author of algorithm is Peter Morris (UK) (Faststrings unit from www.torry.ru). }
4376: function PosStrLast(const FindString, SourceString: string): Integer;
4377: {finds the last occurrence}
4378: function LastPosChar(const FindChar: Char; SourceString: string): Integer;
4379: function PosText(const FindString, SourceString: string; StartPos: Integer = 1): Integer;
4380: { PosText searches the first occurrence of a substring FindString in a string
4381:   given by SourceString without case sensitivity (upper and lower case characters are not differed). This
4382:   function returns the index value of the first character of a specified substring from which it occurs in a
4383:   given string starting with Start
4382: function PosTextLast(const FindString, SourceString: string): Integer;
4383: {finds the last occurrence}
4384: function NameValuesToXML(const AText: string): string;
4385: {$IFDEF MSWINDOWS}
4386: procedure LoadResourceFile(AFile: string; MemStream: TMemoryStream);
4387: {$ENDIF MSWINDOWS}
4388: procedure DirFiles(const ADir, AMask: string; AFileList: TStringList);
4389: procedure RecurseDirFiles(const ADir: string; var AFileList: TStringList);
4390: procedure RecurseDirProgs(const ADir: string; var AFileList: TStringList);
4391: procedure SaveString(const AFile, AText: string);
4392: procedure SaveStringasFile( const AFile, AText : string)
4393: function LoadStringJ(const AFile: string): string;
4394: Function LoadStringOfFile( const AFile : string) : string
4395: Procedure SaveStringToFile( const AFile, AText : string)
4396: Function LoadStringFromFile( const AFile : string) : string
4397: function HexToColor(const AText: string): TColor;
4398: function UppercaseHTMLTags(const AText: string): string;
4399: function LowercaseHTMLTags(const AText: string): string;
4400: procedure GetHTMLAnchors(const AFile: string; AList: TStringList);
4401: function RelativePath(const ASrc, ADst: string): string;
4402: function GetToken(var Start: Integer; const SourceText: string): string;
4403: function PosNonSpace(Start: Integer; const SourceText: string): Integer;
4404: function PosEscaped(Start: Integer; const SourceText, FindText: string; EscapeChar: Char): Integer;
4405: function DeleteEscaped(const SourceText: string; EscapeChar: Char): string;
4406: function BeginOfAttribute(Start: Integer; const SourceText: string): Integer;
4407: // parses the beginning of an attribute: space + alpha character
4408: function ParseAttribute(var Start:Integer; const SourceText:string; var AName,AValue:string): Boolean;
4409: //parses a name="value" attrib from Start; returns 0 when not found or else the position behind attribute
4410: procedure ParseAttributes(const SourceText: string; Attributes: TStrings);
4411: // parses all name=value attributes to the attributes TStringList
4412: function HasStrValue(const AText, AName: string; var AValue: string): Boolean;
4413: // checks if a name="value" pair exists and returns any value
4414: function GetStrValue(const AText, AName, ADefault: string): string;
4415: // retrieves string value from a line like:
4416: // name="jan verhoeven" email="jan1 dott verhoeven att wxs dott nl"
4417: // returns ADefault when not found
4418: function GetHTMLColorValue(const AText, AName: string; ADefault: TColor): TColor;
4419: // same for a color
4420: function GetIntValue(const AText, AName: string; ADefault: Integer): Integer;
4421: // same for an Integer
4422: function GetFloatValue(const AText, AName: string; ADefault: Extended): Extended;
4423: // same for a float
4424: function GetBoolValue(const AText, AName: string): Boolean;
4425: // same for Boolean but without default
4426: function GetValue(const AText, AName: string): string;
4427: //retrieves string value from a line like: name="jan verhoeven" email="jan1 verhoeven att wxs dott nl"
4428: procedure SetValue(var AText: string; const AName, AValue: string);
4429: // sets a string value in a line
4430: procedure DeleteValue(var AText: string; const AName: string);
4431: // deletes a AName="value" pair from AText
4432: procedure GetNames(AText: string; AList: TStringList);
4433: // get a list of names from a string with name="value" pairs
4434: function GetHTMLColor(AColor: TColor): string;
4435: // converts a color value to the HTML hex value
4436: function BackPosStr(Start: Integer; const FindString, SourceString: string): Integer;
4437: // finds a string backward case sensitive
4438: function BackPostText(Start: Integer; const FindString, SourceString: string): Integer;
4439: // finds a string backward case insensitive
4440: function PosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string);
4441: var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4442: // finds a text range, e.g. <TD>....</TD> case sensitive
4443: function PosRangeText(Start: Integer; const HeadString, TailString, SourceString: string);
4444: var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4445: // finds a text range, e.g. <TD>....</td> case insensitive
4446: function BackPosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string);
4447: var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4448: // finds a text range backward, e.g. <TD>....</TD> case sensitive
4449: function BackPosRangeText(Start: Integer; const HeadString, TailString, SourceString: string);

```

```

4450:  var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4451: // finds a text range backward, e.g. <TD>....</td> case insensitive
4452: function PosTag(Start: Integer; SourceString: string; var RangeBegin: Integer;
4453: var RangeEnd: Integer): Boolean;
4454: // finds a XML tag: <....>
4455: function InnerTag(Start: Integer; const HeadString, TailString, SourceString: string;
4456: var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4457: // finds the innerText between opening and closing tags
4458: function Easter(NYear: Integer): TDateTime;
4459: // returns the easter date of a year.
4460: function GetWeekNumber(Today: TDateTime): string;
4461: // gets a datecode. Returns year and weeknumber in format: YYWW
4462: function ParseNumber(const S: string): Integer;
4463: // parse number returns the last position, starting from 1
4464: function ParseDate(const S: string): Integer;
4465: // parse a SQL style data string from positions 1,
4466: // starts and ends with #
4467:
4468: *****unit JvJCLUtils;*****
4469:
4470: function VarIsInt(Value: Variant): Boolean;
4471: // VarIsInt returns VarIsOrdinal-[varBoolean]
4472: { PosIdx returns the index of the first appearance of SubStr in Str. The search starts at index "Index". }
4473: function PosIdx(const SubStr, S: string; Index: Integer = 0): Integer;
4474: function PosIdxW(const SubStr, S: WideString; Index: Integer = 0): Integer;
4475: function PosLastCharIdx(Ch: Char; const S: string; Index: Integer = 0): Integer;
4476: { GetWordOnPos returns Word from string, S, on the cursor position, P}
4477: function GetWordOnPos(const S: string; const P: Integer): string;
4478: function GetWordOnPosW(const S: WideString; const P: Integer): WideString;
4479: function GetWordOnPos2(const S: string; P: Integer; var iBeg, iEnd: Integer): string;
4480: function GetWordOnPos2W(const S: WideString; P: Integer; var iBeg, iEnd: Integer): WideString;
4481: { GetWordOnPosEx working like GetWordOnPos function, but
4482:   also returns Word position in iBeg, iEnd variables }
4483: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4484: function GetWordOnPosExW(const S: WideString; const P: Integer; var iBeg, iEnd: Integer): WideString;
4485: function GetNextWordPosEx(const Text: string; StartIndex: Integer; var iBeg, iEnd: Integer): string;
4486: function GetNextWordPosExW(const Text: WideString; StartIndex: Integer; var iBeg, iEnd: Integer): WideString;
4487: procedure GetEndPosCaret(const Text: string; CaretX, CaretY: Integer; var X, Y: Integer);
4488: { GetEndPosCaret returns the caret position of the last char. For the position
4489:   after the last char of Text you must add 1 to the returned X value. }
4490: procedure GetEndPosCaretW(const Text: WideString; CaretX, CaretY: Integer; var X, Y: Integer);
4491: { GetEndPosCaret returns the caret position of the last char. For the position
4492:   after the last char of Text you must add 1 to the returned X value. }
4493: { SubStrBySeparator returns substring from string, S, separated with Separator string}
4494: function SubStrBySeparator(const S:string;const Index:Integer;const
        Separator:string;startIndex:Int=1):string;
4495: function SubStrBySeparatorW(const S:WideString;const Index:Int;const
        Separator:WideString;startIndex:Int:WideString;
4496: { SubStrEnd same to previous function but Index numerated from the end of string }
4497: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4498: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
4499: function SubWord(P: PChar; var P2: PChar): string;
4500: function CurrencyByWord(Value: Currency): string;
4501: { GetLineByPos returns the Line number, there the symbol Pos is pointed. Lines separated with #13 symbol }
4502: function GetLineByPos(const S: string; const Pos: Integer): Integer;
4503: { GetXYByPos is same as GetLineByPos, but returns X position in line as well}
4504: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4505: procedure GetXYByPosW(const S: WideString; const Pos: Integer; var X, Y: Integer);
4506: { ReplaceString searches for all substrings, OldPattern,
4507:   in a string, S, and replaces them with NewPattern }
4508: function ReplaceString(S: string; const OldPattern,NewPattern: string; startIndex:Integer = 1):string;
4509: function ReplaceStringW(S: WideString; const OldPattern,NewPattern:
        WideString;startIndex:Integer=1):WideString;
4510: { ConcatSep concatenate S1 and S2 strings with Separator. if S = '' then separator not included }
4511: function ConcatSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
        SUPPORTS_INLINE}
4512: { ConcatLeftSep is same to previous function, but   strings concatenate right to left }
4513: function ConcatLeftSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
        SUPPORTS_INLINE}
4514:
4515: { Next 4 function for russian chars transliterating.
4516:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes suck }
4517: procedure Dos2Win(var S: AnsiString);
4518: procedure Win2Dos(var S: AnsiString);
4519: function Dos2WinRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4520: function Win2DOSRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4521: function Win2Koi(const S: AnsiString): AnsiString;
4522: { FillString fills the string Buffer with Count Chars }
4523: procedure FillString(var Buffer: string; Count: Integer; const Value: Char); overload;
4524: procedure FillString(var Buffer: string; startIndex, Count: Integer; const Value: Char); overload;
4525: { MoveString copies Count Chars from Source to Dest }
4526: procedure MoveString(const Source: string; var Dest: string; Count: Integer); {$IFDEF SUPPORTS_INLINE}
        inline; {$ENDIF SUPPORTS_INLINE} overload;
4527: procedure MoveString(const Source: string; SrcStartIdx: Integer; var Dest: string;
        DstStartIdx: Integer;Count: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE} overload;
4528: { FillWideChar fills Buffer with Count WideChars (2 Bytes) }
4529: procedure FillWideChar(var Buffer: WideString; Count: Integer; const Value: WideChar);
4530: { MoveWideChar copies Count WideChars from Source to Dest }
4531: procedure MoveWideChar(const Source: string; var Dest: WideString; Count: Integer); {$IFDEF SUPPORTS_INLINE}
        inline; {$ENDIF SUPPORTS_INLINE}

```

```

4533: { FillNativeChar fills Buffer with Count NativeChars }
4534: procedure FillNativeChar(var Buffer; Count: Integer; const Value: Char); // D2009 internal error {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4535: { MoveWideChar copies Count WideChars from Source to Dest }
4536: procedure MoveNativeChar(const Source; var Dest; Count: Integer); // D2009 internal error {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4537: { IsSubString() compares the sub string to the string. Indices are 1th based. }
4538: function IsSubString(const S: string; StartIndex: Integer; const SubStr: string): Boolean;
4539: { Spaces returns string consists on N space chars }
4540: function Spaces(const N: Integer): string;
4541: { AddSpaces adds spaces to string S, if its Length is smaller than N }
4542: function AddSpaces(const S: string; const N: Integer): string;
4543: function SpacesW(const N: Integer): WideString;
4544: function AddSpacesW(const S: WideString; const N: Integer): WideString;
4545: { function LastDateRUS for russian users only }
4546: { returns date relative to current date: 'àâà àÿ àçàä' }
4547: function LastDateRUS(const Dat: TDateTime): string;
4548: { CurrencyToStr format Currency, Cur, using ffcurrency float format }
4549: function CurrencyToStr(const Cur: Currency): string;
4550: { HasChar returns True, if Char, Ch, contains in string, S }
4551: function HasChar(const Ch: Char; const S: string): Boolean;
4552: function HasCharW(const Ch: WideChar; const S: WideString): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4553: function HasAnyChar(const Chars: string; const S: string): Boolean;
4554: {$IFNDEF COMPILER12_UP}
4555: function CharInSet(const Ch: AnsiChar; const SetOfChar: TSysCharSet): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4556: {$ENDIF ~COMPILER12_UP}
4557: function CharInSetW(const Ch: WideChar; const SetOfChar: TSysCharSet): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4558: function CountOfChar(const Ch: Char; const S: string): Integer;
4559: function DefStr(const S: string; Default: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4560: { StrLICompW2 is a faster replacement for JclUnicode.StrLICompW }
4561: function StrLICompW2(S1, S2: PWideChar; MaxLen: Integer): Integer;
4562: function StrPosW(S, SubStr: PWideChar): PWideChar;
4563: function StrLenW(S: PWideChar): Integer;
4564: function TrimW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4565: function TrimLeftW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4566: function TrimRightW(const S: WideString): WideString; inline; {$ENDIF SUPPORTS_INLINE}
4567: TPixelFormat', '(pfDevice, pf1bit, pf4bit, pf8bit, pf24bit)
4568: TMappingMethod', '(mmHistogram, mmQuantize, mmTrunc784, mmTrunc666, mmTripel, mmGrayscale )
4569: Function GetBitmapPixelFormat(Bitmap : TBitmap) : TPixelFormat
4570: Function GetPaletteBitmapFormat(Bitmap : TBitmap) : TPixelFormat
4571: Procedure SetBitmapPixelFormat(Bitmap : TBitmap; PixelFormat: TPixelFormat; Method: TMappingMethod)
4572: Function BitmapToMemoryStream(Bitmap:TBitmap;PixelFormat:TPixelFormat;Method:TMappingMethod):TMemoryStream;
4573: Procedure GrayscaleBitmap(Bitmap : TBitmap)
4574: Function BitmapToMemory(Bitmap : TBitmap; Colors : Integer) : TStream
4575: Procedure SaveBitmapToFile( const Filename: string; Bitmap : TBitmap; Colors : Integer)
4576: Function ScreenPixelFormat : TPixelFormat
4577: Function ScreenColorCount : Integer
4578: Procedure TileImage( Canvas : TCanvas; Rect : TRect; Image : TGraphic)
4579: Function ZoomImage( ImageW, ImageH, MaxW, MaxH : Integer; Stretch : Boolean) : TPoint
4580: // SIRegister_TJvGradient(CL);
4581:
4582: {***** files routines}
4583: procedure SetDelimitedText(List: TStringList; const Text: string; Delimiter: Char);
4584: const
4585: {$IFDEF MSWINDOWS}
4586: DefaultCaseSensitivity = False;
4587: {$ENDIF MSWINDOWS}
4588: {$IFDEF UNIX}
4589: DefaultCaseSensitivity = True;
4590: {$ENDIF UNIX}
4591: { GenTempFileName returns temporary file name on
4592:   drive, there FileName is placed }
4593: function GenTempFileName(FileName: string): string;
4594: { GenTempFileNameExt same to previous function, but
4595:   returning filename has given extension, FileExt }
4596: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
4597: { ClearDir clears folder Dir }
4598: function ClearDir(const Dir: string): Boolean;
4599: { DeleteDir clears and than delete folder Dir }
4600: function DeleteDir(const Dir: string): Boolean;
4601: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4602: function FileEquMask(FileName, Mask: TFileName; CaseSensitive: Boolean=DefaultCaseSensitivity): Boolean;
4603: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4604:   Masks must be separated with SepPath (MSW: ';' / UNIX: ':') }
4605: function FileEquMasks(FileName, Masks: TFileName;
4606: CaseSensitive: Boolean = DefaultCaseSensitivity): Boolean;
4607: function DeleteFiles(const Folder: TFileName; const Masks: string): Boolean;
4608: {$IFDEF MSWINDOWS}
4609: { LZFileExpand expand file, FileSource,
4610:   into FileDest. Given file must be compressed, using MS Compress program }
4611: function LZFileExpand(const FileSource, FileDest: string): Boolean;
4612: {$ENDIF MSWINDOWS}
4613: { FileGetInfo finds SearchRec record for specified file attributes}
4614: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
4615: { HasSubFolder returns True, if folder APath contains other folders }
4616: function HasSubFolder(APath: TFileName): Boolean;

```

```

4617: { IsEmptyFolder returns True, if there are no files or
4618:   folders in given folder, APath}
4619: function IsEmptyFolder(APath: TFileName): Boolean;
4620: { AddSlash returns string with added slash Char to Dir parameter, if needed }
4621: function AddSlash(const Dir: TFileName): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4622: { AddPath returns FileName with Path, if FileName not contain any path }
4623: function AddPath(const FileName, Path: TFileName): TFileName;
4624: function AddPaths(const PathList, Path: string): string;
4625: function ParentPath(const Path: TFileName): TFileName;
4626: function FindInPath(const FileName, PathList: string): TFileName;
4627: { DeleteReadOnlyFile clears R/O file attribute and delete file }
4628: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
4629: { HasParam returns True, if program running with specified parameter, Param }
4630: function HasParam(const Param: string): Boolean;
4631: function HasSwitch(const Param: string): Boolean;
4632: function Switch(const Param: string): string;
4633: { ExePath returns ExtractFilePath(ParamStr(0)) }
4634: function ExePath: TFileName; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4635: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
4636: //function FileTimeToDateTime(const FT: TFileTime): TDateTime;
4637: procedure FiletimeToDosDateTimeDWord(const FT: TFileTime; out Dft: DWORD);
4638: function MakeValidFileName(const FileName: TFileName; ReplaceBadChar: Char): TFileName;
4639: {**** Graphic routines }
4640: { IsTTFFontSelected returns True, if True Type font is selected in specified device context }
4641: function IsTTFFontSelected(const DC: HDC): Boolean;
4642: function KeyPressed(VK: Integer): Boolean;
4643: Function isKeypressed: boolean; //true if key on memo2 (shell output) is pressed
4644: { TrueInflateRect inflates rect in other method, than InflateRect API function }
4645: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
4646: {**** Color routines }
4647: procedure RGBtoHSV(R, G, B: Integer; var H, S, V: Integer);
4648: function RGBtoBGR(Value: Cardinal): Cardinal;
4649: //function ColorToPrettyName(Value: TColor): string;
4650: //function PrettyNameToColor(const Value: string): TColor;
4651: {**** other routines }
4652: procedure SwapInt(var Int1, Int2: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4653: function IntPower(Base, Exponent: Integer): Integer;
4654: function ChangeTopException(E: TObject): TObject; // Linux version writes error message to ErrOutput
4655: function StrToBool(const S: string): Boolean;
4656: function Var2Type(V: Variant; const DestVarType: Integer): Variant;
4657: function VarToInt(V: Variant): Integer;
4658: function VarToFloat(V: Variant): Double;
4659: { following functions are not documented because they not work properly sometimes, so do not use them }
4660: // (rom) ReplaceStrings1, GetSubStr removed
4661: function GetLongFileName(const FileName: string): string;
4662: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
4663: function GetParameter: string;
4664: function GetComputerID: string;
4665: function GetComputerName: string;
4666: {**** string routines }
4667: { ReplaceAllStrings searches for all substrings, Words,
4668:   in a string, S, and replaces them with Frases with the same Index. }
4669: function ReplaceAllStrings(const S: string; Words, Frases: TStrings): string;
4670: { ReplaceStrings searches the Word in a string, S, on PosBeg position,
4671:   in the list, Words, and if founds, replaces this Word with string from another list,Frases, with the
4672:   same Index, and then update NewSelStart variable }
4673: function ReplaceStrings(const S:string;PosBeg,Len:Int;Words,Frases:TStrings;var NewSelStart:Int):string;
4674: { CountOfLines calculates the lines count in a string, S,
4675:   each line must be separated from another with CrLf sequence }
4676: function CountOfLines(const S: string): Integer;
4677: { DeleteLines deletes all lines from strings which in the words, words.
4678:   The word of will be deleted from strings. }
4679: procedure DeleteOfLines(Ss: TStrings; const Words: array of string);
4680: { DeleteEmptyLines deletes all empty lines from strings, Ss.
4681:   Lines contained only spaces also deletes. }
4682: { SQLAddWhere ades or modifies existing where-statement, where,
4683:   to the strings, SQL. Note: If strings SQL already contains where-statement,
4684:   it must be started on the begining of any line }
4685: procedure SQLAddWhere(SQL: TStrings; const Where: string);
4686: {**** files routines - }
4687: {$IFDEF MSWINDOWS}
4688: { ResSaveToFile save resource named as Name with Typ type into file FileName.
4689:   Resource can be compressed using MS Compress program}
4690: function ResSaveToFile(const Typ,Name:string; const Compressed: Boolean; const FileName: string): Boolean;
4691: function ResSaveToFileEx(Instance:HINST;Typ,Name:PChar;const Compressed:Boolean;const FileName:string): Boolean;
4692: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
4693: {$ENDIF MSWINDOWS}
4694: { IniReadSection read section, Section, from ini-file,
4695:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
4696:   Note: TIniFile.ReadSection function reads only strings with '=' symbol.}
4697: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
4698: { LoadTextFile load text file, FileName, into string }
4699: function LoadTextFile(const FileName: TFileName): string;
4700: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4701: { ReadFolder reads files list from disk folder,Folder,that are equal to mask, Mask,into strings, FileList}
4702: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
4703: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;

```

```

4704: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
4705: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
4706: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
4707: function RATextOutEx(Canvas: TCanvas; const R, RClip: TRect; const S: string; const CalcHeight: Boolean): Integer;
4708: { RATextCalcHeight calculate needed height for
4709:   correct output, using RATextOut or RATextOutEx functions }
4710: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
4711: { Cinema draws some visual effect }
4712: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
4713: { Roughed fills rect with special 3D pattern }
4714: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
4715: { BitmapFromBitmap creates new small bitmap from part of source bitmap, SrcBitmap, with specified width
4716:   and height, AWidth, AHeight and placed on a specified Index, Index in the source bitmap }
4717: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
4718: function TextWidth(const AStr: string): Integer;
4719: { TextHeight calculate text height for writing using standard desktop font }
4720: function TextHeight(const AStr: string): Integer;
4721: procedure SetChildPropOrd(Owner: TComponent; const PropName: string; Value: Longint);
4722: procedure Error(const Msg: string);
4723: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
4724:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
4725: {example Text parameter:'Item 1<b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green<c:blue>blue</i>' }
4726: function ItemHtDraw(Canvas: TCanvas; Rect: TRect;
4727:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): string;
4728: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;
4729:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): Integer;
4730: function ItemHtPlain(const Text: string): string;
4731: { ClearList - clears list of TObject }
4732: procedure ClearList(List: TList);
4733: procedure MemStreamToClipboard(MemStream: TMemoryStream; const Format: Word);
4734: procedure ClipboardToMemStream(MemStream: TMemoryStream; const Format: Word);
4735: { RTTI support }
4736: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
4737: function GetPropStr(Obj: TObject; const PropName: string): string;
4738: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
4739: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
4740: procedure PrepareIniSection(Ss: TStrings);
4741: { following functions are not documented because they are don't work properly, so don't use them }
4742: // (rom) from JvBandWindows to make it obsolete
4743: function PointL(const X, Y: Longint): TPointL; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4744: // (rom) from JvBandUtils to make it obsolete
4745: function iif(const Test: Boolean; const ATrue, AFalse: Variant): Variant;
4746: procedure CopyIconToClipboard(Icon: TIcon; BackColor: TColor);
4747: function CreateIconFromClipboard: TIcon;
4748: { begin JvIconClipboardUtils } { Icon clipboard routines }
4749: function CF_ICON: Word;
4750: procedure AssignClipboardIcon(Icon: TIcon);
4751: { Real-size icons support routines (32-bit only) }
4752: procedure GetIconSize(Icon: HICON; var W, H: Integer);
4753: function CreateRealSizeIcon(Icon: TIcon): HICON;
4754: procedure DrawRealSizeIcon(Canvas: TCanvas; Icon: TIcon; X, Y: Integer);
4755: {end JvIconClipboardUtils }
4756: function CreateScreenCompatibleDC: HDC;
4757: function InvalidateRect(hWnd: HWND; const lpRect: TRect; bErase: BOOL): BOOL; overload; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF}
4758: function InvalidateRect(hWnd: HWND; lpRect: PRect; bErase: BOOL): BOOL; overload;
4759: { begin JvRLE } // (rom) changed API for inclusion in JCL
4760: procedure RleCompressTo(InStream, OutStream: TStream);
4761: procedure RleDecompressTo(InStream, OutStream: TStream);
4762: procedure RleCompress(Stream: TStream);
4763: procedure RleDecompress(Stream: TStream);
4764: { end JvRLE } { begin JvDateUtil }
4765: function CurrentYear: Word; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4766: function IsLeapYear(AYear: Integer): Boolean;
4767: function DaysInAMonth(const AYear, AMonth: Word): Word;
4768: function DaysPerMonth(AYear, AMonth: Integer): Integer;
4769: function FirstDayOfPrevMonth: TDateTime;
4770: function LastDayOfPrevMonth: TDateTime;
4771: function FirstDayOfNextMonth: TDateTime;
4772: function ExtractDay(ADate: TDateTime): Word;
4773: function ExtractMonth(ADate: TDateTime): Word;
4774: function ExtractYear(ADate: TDateTime): Word;
4775: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
4776: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime; inline; {$IFDEF SUPPORTS_INLINE}
4777: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
4778: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
4779: function ValidDate(ADate: TDateTime): Boolean;
4780: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
4781: function MonthsBetween(Date1, Date2: TDateTime): Double;
4782: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
4783: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
4784: function DaysBetween(Date1, Date2: TDateTime): Longint;
4785: { The same as previous but if Date2 < Date1 result = 0 }
4786: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
4787: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
4788: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
4789: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
4790: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;

```

```

4791: function CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
4792: { String to date conversions }
4793: function GetDateOrder(const DateFormat: string): TDateOrder;
4794: function MonthFromName(const S: string; MaxLen: Byte): Byte;
4795: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
4796: function StrToDateFmt(const DateFormat, S: string): TDateTime;
4797: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
4798: //function DefDateFormat(AFourDigitYear: Boolean): string;
4799: //function DefDateMask(BlanksChar: Char; AFourDigitYear: Boolean): string;
4800: function FormatLongDate(Value: TDateTime): string;
4801: function FormatLongDateTime(Value: TDateTime): string;
4802: { end JvDateUtil }
4803: function BufToBinStr(Buf: Pointer; BufSize: Integer): string;
4804: function BinStrToBuf(Value: string; Buf: Pointer; BufSize: Integer): Integer;
4805: { begin JvStrUtils } { ** Common string handling routines ** }
4806: {$IFDEF UNIX}
4807: function iconversion(InP: PAnsiChar; OutP: Pointer; InBytes, OutBytes: Cardinal;
4808: const ToCode, FromCode: AnsiString): Boolean;
4809: function iconvString(const S, ToCode, FromCode: AnsiString): string;
4810: function iconvWideString(const S: WideString; const ToCode, FromCode: AnsiString): WideString;
4811: function OemStrToAnsi(const S: AnsiString): AnsiString;
4812: function AnsiStrToOem(const S: AnsiString): AnsiString;
4813: {$ENDIF UNIX}
4814: function StrToOem(const AnsiStr: AnsiString): AnsiString;
4815: { StrToOem translates a string from the Windows character set into the OEM character set. }
4816: function OemToAnsiStr(const OemStr: AnsiString): AnsiString;
4817: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4818: function IsEmptyStr(const S: string; const EmptyChars: TSysCharSet): Boolean;
4819: { EmptyStr returns True if the given string contains only character from the EmptyChars. }
4820: function ReplaceStr(const S, Srch, Replace: string): string;
4821: { Returns string with every occurrence of Srch string replaced with Replace string. }
4822: function DelSpace(const S: string): string;
4823: { DelSpace return a string with all white spaces removed. }
4824: function DelChars(const S: string; Chr: Char): string;
4825: { DelChars return a string with all Chr characters removed. }
4826: function DelBSpace(const S: string): string;
4827: { DelBSpace trims leading spaces from the given string. }
4828: function DelESpace(const S: string): string;
4829: { DelESpace trims trailing spaces from the given string. }
4830: function DelRSpace(const S: string): string;
4831: { DelRSpace trims leading and trailing spaces from the given string. }
4832: function DelSpace1(const S: string): string;
4833: { DelSpace1 return a string with all non-single white spaces removed. }
4834: function Tab2Space(const S: string; Numb: Byte): string;
4835: { Tab2Space converts any tabulation character in the given string to the
4836: Numb spaces characters. }
4837: function NPos(const C: string; S: string; N: Integer): Integer;
4838: { NPos searches for a N-th position of substring C in a given string. }
4839: function MakeStr(C: Char; N: Integer): string; overload;
4840: {$IFNDEF COMPILER12_UP}
4841: function MakeStr(C: WideChar; N: Integer): WideString; overload;
4842: {$ENDIF !COMPILER12_UP}
4843: function MS(C: Char; N: Integer): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4844: { MakeStr return a string of length N filled with character C. }
4845: function AddChar(C: Char; const S: string; N: Integer): string;
4846: { AddChar return a string left-padded to length N with characters C. }
4847: function AddCharR(C: Char; const S: string; N: Integer): string;
4848: { AddCharR return a string right-padded to length N with characters C. }
4849: function LeftStr(const S: string; N: Integer): string;
4850: { LeftStr return a string right-padded to length N with blanks. }
4851: function RightStr(const S: string; N: Integer): string;
4852: { RightStr return a string left-padded to length N with blanks. }
4853: function CenterStr(const S: string; Len: Integer): string;
4854: { CenterStr centers the characters in the string based upon the Len specified. }
4855: function CompStr(const S1, S2: string): Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4856: { CompStr compares S1 to S2, with case-sensitivity. The return value is
4857: -1 if S1 < S2, 0 if S1 = S2, or 1 if S1 > S2. }
4858: function CompText(const S1, S2: string): Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4859: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4860: function Copy2Symb(const S: string; Symb: Char): string;
4861: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4862: function Copy2SymbDel(var S: string; Symb: Char): string;
4863: { Copy2SymbDel returns a substring of a string S from begining to first
4864: character Symb and removes this substring from S. }
4865: function Copy2Space(const S: string): string;
4866: { Copy2Space returns a substring of a string S from begining to first white space. }
4867: function Copy2SpaceDel(var S: string): string;
4868: { Copy2SpaceDel returns a substring of a string S from begining to first
4869: white space and removes this substring from S. }
4870: function AnsiProperCase(const S: string; const WordDelims: TSysCharSet): string;
4871: { Returns string, with the first letter of each word in uppercase,
4872: all other letters in lowercase. Words are delimited by WordDelims. }
4873: function WordCount(const S: string; const WordDelims: TSysCharSet): Integer;
4874: { WordCount given a set of word delimiters, returns number of words in S. }
4875: function WordPosition(const N: Integer; const S: string; const WordDelims: TSysCharSet): Integer;
4876: { Given a set of word delimiters, returns start position of N'th word in S. }
4877: function ExtractWord(N: Integer; const S: string; const WordDelims: TSysCharSet): string;
4878: function ExtractWordPos(N: Integer; const S: string; const WordDelims: TSysCharSet; var Pos: Integer): string;
4879: function ExtractDelimited(N: Integer; const S: string; const Delims: TSysCharSet): string;

```

```

4880: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4881:   delimiters, return the N'th word in S. }
4882: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TSysCharSet): string;
4883: { ExtractSubstr given a set of word delimiters, returns the substring from S,
4884:   that started from position Pos. }
4885: function IsWordPresent(const W, S: string; const WordDelims: TSysCharSet): Boolean;
4886: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4887: function QuotedString(const S: string; Quote: Char): string;
4888: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4889: function ExtractQuotedString(const S: string; Quote: Char): string;
4890: { ExtractQuotedString removes the Quote characters from the beginning and
4891:   end of a quoted string, and reduces pairs of Quote characters within quoted string to single character. }
4892: function FindPart(const HelpWilds, InputStr: string): Integer;
4893: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4894: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4895: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4896: function XorString(const Key, Src: ShortString): ShortString;
4897: function XorEncode(const Key, Source: string): string;
4898: function XorDecode(const Key, Source: string): string;
4899: { ** Command line routines ** }
4900: function GetCmdLineArg(const Switch: string; ASwitchChars: TSysCharSet): string;
4901: { ** Numeric string handling routines ** }
4902: function Numb2USA(const S: string): string;
4903: { Numb2USA converts numeric string S to USA-format. }
4904: function Dec2Hex(N: Longint; A: Byte): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4905: { Dec2Hex converts the given value to a hexadecimal string representation
4906:   with the minimum number of digits (A) specified. }
4907: function Hex2Dec(const S: string): Longint;
4908: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4909: function Dec2Numb(N: Int64; A, B: Byte): string;
4910: { Dec2Numb converts the given value to a string representation with the
4911:   base equal to B and with the minimum number of digits (A) specified. }
4912: function Numb2Dec(S: string; B: Byte): Int64;
4913: { Numb2Dec converts the given B-based numeric string to the corresponding
4914:   integer value. }
4915: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4916: { IntToBin converts the given value to a binary string representation
4917:   with the minimum number of digits specified. }
4918: function IntToRoman(Value: Longint): string;
4919: { IntToRoman converts the given value to a roman numeric string representation. }
4920: function RomanToInt(const S: string): Longint;
4921: { RomanToInt converts the given string to an integer value. If the string
4922:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4923: function FindNotBlankCharPos(const S: string): Integer;
4924: function FindNotBlankCharPosW(const S: WideString): Integer;
4925: function AnsiChangeCase(const S: string): string;
4926: function WideChangeCase(const S: string): string;
4927: function StartsText(const SubStr, S: string): Boolean;
4928: function EndsText(const SubStr, S: string): Boolean;
4929: function DequotedStr(const S: string; QuoteChar: Char = ''): string;
4930: function AnsiDequotedStr(const S: string; AQuote: Char): string; //follow Delphi 2009's "Ansi" prefix
4931: {end JvStrUtils}
4932: {$IFDEF UNIX}
4933: function GetTempFileName(const Prefix: AnsiString): AnsiString;
4934: {$ENDIF UNIX}
4935: { begin JvFileUtil }
4936: function FileDateTime(const FileName: string): TDateTime;
4937: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4938: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4939: function NormalDir(const DirName: string): string;
4940: function RemoveBackSlash(const DirName: string): string; // only for Windows/DOS Paths
4941: function ValidfileName(const FileName: string): Boolean;
4942: {$IFDEF MSWINDOWS}
4943: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4944: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4945: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4946: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4947: {$ENDIF MSWINDOWS}
4948: function GetWindowsDir: string;
4949: function GetSystemDir: string;
4950: function ShortToLongFileName(const ShortName: string): string;
4951: function LongToShortFileName(const LongName: string): string;
4952: function ShortToLongPath(const ShortName: string): string;
4953: function LongToShortPath(const LongName: string): string;
4954: {$IFDEF MSWINDOWS}
4955: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4956: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4957: {$ENDIF MSWINDOWS}
4958: { end JvFileUtil }
4959: // Works like PtInRect but includes all edges in comparision
4960: function PtInRectInclusive(R: TRect; Pt: TPoint): Boolean;
4961: // Works like PtInRect but excludes all edges from comparision
4962: function PtInRectExclusive(R: TRect; Pt: TPoint): Boolean;
4963: function FourDigitYear: Boolean; {$IFDEF SUPPORTS_DEPRECATED} deprecated; {$ENDIF}
4964: function IsFourDigitYear: Boolean;
4965: { moved from JvJVCLUTils }
4966: //Open an object with the shell (url or something like that)
4967: function OpenObject(const Value: string): Boolean; overload;
4968: function OpenObject(Value: PChar): Boolean; overload;

```

```

4969: {$IFDEF MSWINDOWS}
4970: //Raise the last Exception
4971: procedure RaiseLastWin32; overload;
4972: procedure RaiseLastWin32(const Text: string); overload;
4973: //Raise the last Exception with a small comment from your part { GetFileVersion returns the most
4974: // significant 32 bits of a file's binary version number. Typically, this includes the major and minor
4975: // version placed together in one 32-bit Integer. I
4976: function GetFileVersion(const AFileName: string): Cardinal;
4977: {$EXTERNALSYM GetFileVersion}
4978: function GetShellVersion: Cardinal;
4979: {$EXTERNALSYM GetShellVersion}
4980: // CD functions on HW
4981: procedure OpenCdDrive;
4982: procedure CloseCdDrive;
4983: // returns True if Drive is accessible
4984: function DiskInDrive(Drive: Char): Boolean;
4985: {$ENDIF MSWINDOWS}
4986: //Same as linux function ;
4987: procedure PError(const Text: string);
4988: // execute a program without waiting
4989: procedure Exec(const FileName, Parameters, Directory: string);
4990: // execute a program and wait for it to finish
4991: function ExecuteAndWait(CmdLine:string;const WorkingDirectory:string;Visibility:Integer=SW_SHOW): Int;
4992: // returns True if this is the first instance of the program that is running
4993: function FirstInstance(const ATitle: string): Boolean;
4994: // restores a window based on it's classname and Caption. Either can be left empty
4995: // to widen the search
4996: procedure RestoreOtherInstance(const MainFormClassName, MainFormCaption: string);
4997: // manipulate the traybar and start button
4998: procedure HideTraybar;
4999: procedure ShowTraybar;
5000: procedure ShowStartButton(Visible: Boolean = True);
5001: // (rom) SC_MONITORPOWER is documented as Win 95 only(rom) better do some testing set monitor functions
5002: procedure MonitorOn;
5003: procedure MonitorOff;
5004: procedure LowPower;
5005: // send a key to the window named AppName
5006: function SendKey(const AppName: string; Key: Char): Boolean;
5007: {$IFDEF MSWINDOWS}
5008: // returns a list of all win currently visible, the Objects property is filled with their window handle
5009: procedure GetVisibleWindows(List: TStrings);
5010: Function GetVisibleWindowsF( List : TStrings):TStrings';
5011: // associates an extension to a specific program
5012: procedure AssociateExtension(const IconPath, ProgramName, Path, Extension: string);
5013: function GetRecentDocs: TStringList;
5014: {$ENDIF MSWINDOWS}
5015: function CharIsMoney(const Ch: Char): Boolean;
5016: //function StrToCurrDef(const Str: string; Def: Currency): Currency;
5017: function IntToExtended(I: Integer): Extended;
5018: { GetChangedText works out the new text given the current cursor pos & the key pressed
5019: It is not very useful in other contexts, but in this unit as it is needed in both MemoEx and TypedEdit }
5020: function GetChangedText(const Text: string; SelStart, SelLength: Integer; Key: Char): string;
5021: function MakeYear4Digit(Year, Pivot: Integer): Integer;
5022: //function StrIsInteger(const S: string): Boolean;
5023: function StrIsFloatMoney(const Ps: string): Boolean;
5024: function StrIsDateTime(const Ps: string): Boolean;
5025: function PreformatDateString(Ps: string): string;
5026: function BooleanToInteger(const B: Boolean): Integer;
5027: function StringToBoolean(const Ps: string): Boolean;
5028: function SafeStrToDate(const Ps: string): TDateTime;
5029: function SafeStrToDate(const Ps: string): TDateTime;
5030: function SafeStrToTime(const Ps: string): TDateTime;
5031: function StrDelete(const psSub, psMain: string): string;
5032: { returns the fractional value of pcValue}
5033: function TimeOnly(pcValue: TDateTime): TTime;
5034: { returns the integral value of pcValue }
5035: function DateOnly(pcValue: TDateTime): TDate;
5036: type TdtKind = (dtkDateOnly, dtkTimeOnly, dtkDateTime);
5037: const { TDateTime value used to signify Null value}
5038: NullEquivalentDate: TDateTime = 0.0;
5039: function DateIsNull(const pdtValue: TDateTime; const pdtKind: TdtKind): Boolean;
5040: // Replacement for Win32Check to avoid platform specific warnings in D6
5041: function OSCheck(RetVal: Boolean): Boolean;
5042: { Shortens a fully qualified Path name so that it can be drawn with a specified length limit.
5043: Same as FileCtrl.MinimizeName in functionality (but not implementation). Included here to
5044: not be forced to use FileCtrl unnecessarily }
5045: function MinimizeFileName(const FileName: string; Canvas: TCanvas; MaxLen: Integer): string;
5046: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
5047: { MinimizeString truncates long string, S, and appends...'symbols,if Length of S is more than MaxLen }
5048: function MinimizeString(const S: string; const MaxLen: Integer): string;
5049: procedure RunDll32Internal(Wnd:THandle; const DLLName,FuncName,CmdLine:string;CmdShow:Integer=SW_SHOWDEFAULT);
5050: { GetDLLVersion loads DLLName, gets a pointer to DLLVersion function and calls it, returning major and
5051: minor version values from the function. Returns False if DLL not loaded or if GetDLLVersion couldn't be
5052: found.}
5053: function GetDLLVersion(const DLLName: string; var pdwMajor, pdwMinor: Integer): Boolean;
5054: {$ENDIF MSWINDOWS}

```

```

5053: procedure ResourceNotFound(ResID: PChar);
5054: function EmptyRect: TRect;
5055: function RectWidth(R: TRect): Integer;
5056: function RectHeight(R: TRect): Integer;
5057: function CompareRect(const R1, R2: TRect): Boolean;
5058: procedure RectNormalize(var R: TRect);
5059: function RectIsSquare(const R: TRect): Boolean;
5060: function RectSquare(var ARect: TRect; AMaxSize: Integer = -1): Boolean;
5061: //If AMaxSize = -1, then auto calc Square's max size
5062: {$IFDEF MSWINDOWS}
5063: procedure FreeUnusedOle;
5064: function GetWindowsVersion: string;
5065: function LoadDLL(const LibName: string): THandle;
5066: function RegisterServer(const ModuleName: string): Boolean;
5067: function UnregisterServer(const ModuleName: string): Boolean;
5068: {$ENDIF MSWINDOWS}
5069: { String routines }
5070: function GetEnvVar(const VarName: string): string;
5071: function AnsiUpperFirstChar(const S: string): string; //follow Delphi 2009's example with "Ansi" prefix
5072: function StringToPChar(var S: string): PChar;
5073: function StrAlloc(const S: string): PChar;
5074: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
5075: function DropT(const S: string): string;
5076: { Memory routines }
5077: function AllocMemo(Size: Longint): Pointer;
5078: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
5079: procedure FreeMemo(var fpBlock: Pointer);
5080: function GetMemoSize(fpBlock: Pointer): Longint;
5081: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
5082: { Manipulate huge pointers routines }
5083: procedure HugeInc(var HugePtr: Pointer; Amount: Longint);
5084: procedure HugeDec(var HugePtr: Pointer; Amount: Longint);
5085: function HugeOffset(HugePtr: Pointer; Amount: Longint): Pointer;
5086: procedure HugeMove(Base: Pointer; Dst, Src, Size: Longint);
5087: procedure HMemCpy(DstPtr, SrcPtr: Pointer; Amount: Longint);
5088: function WindowClassName(Wnd: THandle): string;
5089: procedure SwitchToWindow(Wnd: THandle; Restore: Boolean);
5090: procedure ActivateWindow(Wnd: THandle);
5091: procedure ShowWinNoAnimate(Handle: THandle; CmdShow: Integer);
5092: procedure KillMessage(Wnd: THandle; Msg: Cardinal);
5093: { SetWindowTop put window to top without recreating window }
5094: procedure SetWindowTop(const Handle: THandle; const Top: Boolean);
5095: procedure CenterWindow(Wnd: THandle);
5096: function MakeVariant(const Values: array of Variant): Variant;
5097: { Convert dialog units to pixels and backwards }
5098: {$IFDEF MSWINDOWS}
5099: function DialogUnitsToPixelsX(DlgUnits: Word): Word;
5100: function DialogUnitsToPixelsY(DlgUnits: Word): Word;
5101: function PixelsToDialogUnitsX(PixUnits: Word): Word;
5102: function PixelsToDialogUnitsY(PixUnits: Word): Word;
5103: {$ENDIF MSWINDOWS}
5104: function GetUniqueFileNameInDir(const Path, FileNameMask: string): string;
5105: {$IFDEF BCB}
5106: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): THandle;
5107: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
5108: {$ELSE}
5109: function FindPrevInstance(const MainFormClass, ATitle: string): THandle;
5110: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
5111: {$ENDIF BCB}
5112: {$IFDEF MSWINDOWS}
5113: { BrowseForFolderNative displays Browse For Folder dialog }
5114: function BrowseForFolderNative(const Handle: THandle; const Title: string; var Folder: string): Boolean;
5115: {$ENDIF MSWINDOWS}
5116: procedure AntiAlias(Alias: TBitmap);
5117: procedure AntiAliasRect(Clip: TBitmap; XOrigin, YOrigin, XFinal, YFinal: Integer);
5118: procedure CopyRectDIBits(ACanvas: TCanvas; const DestRect: TRect;
5119: ABitmap: TBitmap; const SourceRect: TRect);
5120: function IsTrueType(const FontName: string): Boolean;
5121: // Removes all non-numeric characters from AValue and returns the resulting string
5122: function TextToValText(const AValue: string): string;
5123: Function ExecRegExpr( const ARegExpr, AInputStr : RegExprString ) : boolean
5124: Procedure SplitRegExpr( const ARegExpr, AInputStr : RegExprString; APieces : TStrings )
5125: Function ReplaceRegExpr( const ARegExpr,AInputStr,
      AReplaceStr:RegExprString;AUseSubstitution:bool):RegExprString;
5126: Function QuoteRegExprMetaChars( const AStr : RegExprString ) : RegExprString
5127: Function RegExprSubExpressions( const ARegExpr:string; ASubExps:TStrings; AExtendedSyntax : boolean ) :
5128:
5129: *****unit uPSI_JvTFUtils;
5130: Function JExtractYear( ADate : TDateTime ) : Word
5131: Function JExtractMonth( ADate : TDateTime ) : Word
5132: Function JExtractDay( ADate : TDateTime ) : Word
5133: Function ExtractHours( ATime : TDateTime ) : Word
5134: Function ExtractMins( ATime : TDateTime ) : Word
5135: Function ExtractSecs( ATime : TDateTime ) : Word
5136: Function ExtractMSecs( ATime : TDateTime ) : Word
5137: Function FirstOfMonth( ADate : TDateTime ) : TDateTime
5138: Function GetDayOfNthDOW( Year, Month, DOW, N : Word ) : Word
5139: Function GetWeeksInMonth( Year, Month : Word; StartOfWeek : Integer ) : Word
5140: Procedure IncBorlDOW( var BorlDOW : Integer; N : Integer )

```

```

5141: Procedure IncDOW( var DOW : TTFFDayOfWeek; N : Integer)
5142: Procedure IncDays( var ADate : TDateTime; N : Integer)
5143: Procedure IncWeeks( var ADate : TDateTime; N : Integer)
5144: Procedure IncMonths( var ADate : TDateTime; N : Integer)
5145: Procedure IncYears( var ADate : TDateTime; N : Integer)
5146: Function EndOfMonth( ADate : TDateTime) : TDateTime
5147: Function IsFirstOfMonth( ADate : TDateTime) : Boolean
5148: Function IsEndOfMonth( ADate : TDateTime) : Boolean
5149: Procedure EnsureMonth( Month : Word)
5150: Procedure EnsureDOW( DOW : Word)
5151: Function EqualDates( D1, D2 : TDateTime) : Boolean
5152: Function Lesser( N1, N2 : Integer) : Integer
5153: Function Greater( N1, N2 : Integer) : Integer
5154: Function GetDivLength( TotalLength, DivCount, DivNum : Integer) : Integer
5155: Function GetDivNum( TotalLength, DivCount, X : Integer) : Integer
5156: Function GetDivStart( TotalLength, DivCount, DivNum : Integer) : Integer
5157: Function DOWToBorl( ADOW : TTFFDayOfWeek) : Integer
5158: Function BorlToDOW( BorlDOW : Integer) : TTFFDayOfWeek
5159: Function DateToDOW( ADate : TDateTime) : TTFFDayOfWeek
5160: Procedure CalcTextPos( HostRect : TRect; var TextLeft, TextTop:Integer; var TextBounds : TRect;
AFont:TFont;AAngle: Integer; HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5161: Procedure DrawAngleText( ACanvas : TCanvas; HostRect : TRect; var TextBounds : TRect; AAngle : Integer;
HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5162: Function JRectWidth( ARect : TRect) : Integer
5163: Function JRectHeight( ARect : TRect) : Integer
5164: Function JEmptyRect : TRect
5165: Function IsClassByName( Obj : TObject; ClassName : string) : Boolean
5166:
5167: procedure SIRegister_MSysUtils(CL: TPSPascalCompiler);
5168: begin
5169:   Procedure HideTaskBarButton( hWindow : HWND)
5170:   Function msLoadStr( ID : Integer) : String
5171:   Function msFormat( fmt : String; params : array of const) : String
5172:   Function msFileExists( const FileName : String) : Boolean
5173:   Function msIntToStr( Int : Int64) : String
5174:   Function msStrPas( const Str : PChar) : String
5175:   Function msRenamefile( const OldName, NewName : String) : Boolean
5176:   Function CutFileName( s : String) : String
5177:   Function GetVersionInfo( var VersionString : String) : DWORD
5178:   Function FormatTime( t : Cardinal) : String
5179:   Function msCreateDir( const Dir : string) : Boolean
5180:   Function SetAutoRun( NeedAutoRun : Boolean; AppName : String) : Boolean
5181:   Function SetTreeLineStyle( const hTV : HWND; dwNewStyle : dword) : DWORD
5182:   Function msStrLen( Str : PChar) : Integer
5183:   Function msDirectoryExists( const Directory : String) : Boolean
5184:   Function GetFolder( hWnd : HWND; RootDir : Integer; Caption : String) : String
5185:   Function SetBlendWindow( hWnd : HWND; AlphaBlend : Byte) : LongBool
5186:   Function EditWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
5187:   Procedure SetEditWndProc( hWnd : HWND; ptr : TObject)
5188:   Function GetTextFromFile( Filename : String) : string
5189:   Function IsTopMost( hWnd : HWND) : Bool // 'LWA_ALPHA','LongWord').SetUIInt( $00000002);
5190:   Function msStrToIntDef( const s : String; const i : Integer) : Integer
5191:   Function msStrToInt( s : String) : Integer
5192:   Function GetItemText( hDlg : THandle; ID : DWORD) : String
5193: end;
5194:
5195: procedure SIRegister_ESBMaths2(CL: TPSPascalCompiler);
5196: begin
5197:   //TDynFloatArray', 'array of Extended
5198:   TDynLWordArray', 'array of LongWord
5199:   TDynLIntArray', 'array of LongInt
5200:   TDynFloatMatrix', 'array of TDynFloatArray
5201:   TDynLWordMatrix', 'array of TDynLWordArray
5202:   TDynLIntMatrix', 'array of TDynLIntArray
5203:   Function SquareAll( const X : TDynFloatArray) : TDynFloatArray
5204:   Function InverseAll( const X : TDynFloatArray) : TDynFloatArray
5205:   Function InAll( const X : TDynFloatArray) : TDynFloatArray
5206:   Function Log10All( const X : TDynFloatArray) : TDynFloatArray
5207:   Function LinearTransform( const X : TDynFloatArray; Offset, Scale : Extended) : TDynFloatArray
5208:   Function AddVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5209:   Function SubVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5210:   Function MultVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5211:   Function DotProduct( const X, Y : TDynFloatArray) : Extended
5212:   Function MNorm( const X : TDynFloatArray) : Extended
5213:   Function MatrixIsRectangular( const X : TDynFloatMatrix) : Boolean
5214:   Procedure MatrixDimensions(const X:TDynFloatMatrix;var Rows,Columns:LongWord;var Rectangular:Boolean);
5215:   Function MatrixIsSquare( const X : TDynFloatMatrix) : Boolean
5216:   Function MatricesSameDimensions( const X, Y : TDynFloatMatrix) : Boolean
5217:   Function AddMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5218:   Procedure AddToMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5219:   Function SubtractMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5220:   Procedure SubtractFromMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5221:   Function MultiplyMatrixByConst( const X : TDynFloatMatrix; const K : Extended) : TDynFloatMatrix
5222:   Procedure MultiplyMatrixByConst2( var X : TDynFloatMatrix; const K : Extended);
5223:   Function MultiplyMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix;
5224:   Function TransposeMatrix( const X : TDynFloatMatrix) : TDynFloatMatrix;
5225:   Function GrandMean( const X : TDynFloatMatrix; var N : LongWord) : Extended
5226: end;
5227:

```

```

5228: procedure SIRegister_ESBMaths(CL: TPSCompiler);
5229: begin
5230:   'ESBMinSingle','Single').setExtended( 1.5e-45);
5231:   'ESBMaxSingle','Single').setExtended( 3.4e+38);
5232:   'ESBMinDouble','Double').setExtended( 5.0e-324);
5233:   'ESBMaxDouble','Double').setExtended( 1.7e+308);
5234:   'ESBMinExtended','Extended').setExtended( 3.6e-4951);
5235:   'ESBMaxExtended','Extended').setExtended( 1.1e+4932);
5236:   'ESBMinCurrency','Currency').SetExtended( - 922337203685477.5807);
5237:   'ESBMaxCurrency','Currency').SetExtended( 922337203685477.5807);
5238:   'ESBSqrt2','Extended').setExtended( 1.4142135623730950488);
5239:   'ESBSqrt3','Extended').setExtended( 1.7320508075688772935);
5240:   'ESBSqrt5','Extended').setExtended( 2.2360679774997896964);
5241:   'ESBSqrt10','Extended').setExtended( 3.1622776601683793320);
5242:   'ESBSqrtPi','Extended').setExtended( 1.77245385090551602729);
5243:   'ESBCbrt2','Extended').setExtended( 1.2599210498948731648);
5244:   'ESBCbrt3','Extended').setExtended( 1.4422495703074083823);
5245:   'ESBCbrt10','Extended').setExtended( 2.1544346900318837219);
5246:   'ESBCbrt100','Extended').setExtended( 4.6415888336127788924);
5247:   'ESBCbrtPi','Extended').setExtended( 1.4645918875615232630);
5248:   'ESBInvSqrt2','Extended').setExtended( 0.70710678118654752440);
5249:   'ESBInvSqrt3','Extended').setExtended( 0.57735026918962576451);
5250:   'ESBInvSqrt5','Extended').setExtended( 0.44721359549995793928);
5251:   'ESBInvSqrtPi','Extended').setExtended( 0.56418958354775628695);
5252:   'ESBInvCbrtPi','Extended').setExtended( 0.68278406325529568147);
5253:   'ESBe','Extended').setExtended( 2.7182818284590452354);
5254:   'ESBe2','Extended').setExtended( 7.3890560989306502272);
5255:   'ESBePi','Extended').setExtended( 23.140692632779269006);
5256:   'ESBePiOn2','Extended').setExtended( 4.8104773809653516555);
5257:   'ESBePiOn4','Extended').setExtended( 2.1932800507380154566);
5258:   'ESBLn2','Extended').setExtended( 0.69314718055994530942);
5259:   'ESBLn10','Extended').setExtended( 2.30258509299404568402);
5260:   'ESBLnP1','Extended').setExtended( 1.14472988584940017414);
5261:   'ESBLog10Base2','Extended').setExtended( 3.3219280948873623478);
5262:   'ESBLog2Base10','Extended').setExtended( 0.30102999566398119521);
5263:   'ESBLog3Base10','Extended').setExtended( 0.47712125471966243730);
5264:   'ESBLogPiBase10','Extended').setExtended( 0.4971498726941339);
5265:   'ESBLogEBase10','Extended').setExtended( 0.43429448190325182765);
5266:   'ESBPi','Extended').setExtended( 3.141592653897932385);
5267:   'ESBInvPi','Extended').setExtended( 3.1830988618379067154e-1);
5268:   'ESBTwoPi','Extended').setExtended( 6.2831853071795864769);
5269:   'ESBThreePi','Extended').setExtended( 9.4247779607693797153);
5270:   'ESBPi2','Extended').setExtended( 9.8696044010893586188);
5271:   'ESBPiToE','Extended').setExtended( 22.459157718361045473);
5272:   'ESBPiOn2','Extended').setExtended( 1.5707963267948966192);
5273:   'ESBPiOn3','Extended').setExtended( 1.0471975511965977462);
5274:   'ESBPiOn4','Extended').setExtended( 0.7853981633974483096);
5275:   'ESBThreePiOn2','Extended').setExtended( 4.7123889803846898577);
5276:   'ESBFourPiOn3','Extended').setExtended( 4.1887902047863909846);
5277:   'ESBTwoToPower63','Extended').setExtended( 9223372036854775808.0);
5278:   'ESBOneRadian','Extended').setExtended( 57.295779513082320877);
5279:   'ESBOneDegree','Extended').setExtended( 1.7453292519943295769E-2);
5280:   'ESBOneMinute','Extended').setExtended( 2.9088820866572159615E-4);
5281:   'ESBOneSecond','Extended').setExtended( 4.8481368110953599359E-6);
5282:   'ESBGamma','Extended').setExtended( 0.57721566490153286061);
5283:   'ESBLnRT2Pi','Extended').setExtended( 9.189385332046727E-1);
5284: //LongWord', 'Cardinal
5285: TBitList', 'Word
5286: Function UMul( const Num1, Num2 : LongWord ) : LongWord
5287: Function UMulDiv2p32( const Num1, Num2 : LongWord ) : LongWord
5288: Function UMulDiv( const Num1, Num2, Divisor : LongWord ) : LongWord
5289: Function UMulMod( const Num1, Num2, Modulus : LongWord ) : LongWord
5290: Function SameFloat( const X1, X2 : Extended ) : Boolean
5291: Function FloatIsZero( const X : Extended ) : Boolean
5292: Function FloatIsPositive( const X : Extended ) : Boolean
5293: Function FloatIsNegative( const X : Extended ) : Boolean
5294: Procedure InclIm( var B : Byte; const Limit : Byte )
5295: Procedure InclImSI( var B : ShortInt; const Limit : ShortInt )
5296: Procedure InclImW( var B : Word; const Limit : Word )
5297: Procedure InclImI( var B : Integer; const Limit : Integer )
5298: Procedure InclImL( var B : LongInt; const Limit : LongInt )
5299: Procedure Declim( var B : Byte; const Limit : Byte )
5300: Procedure DeclimSI( var B : ShortInt; const Limit : ShortInt )
5301: Procedure DeclimW( var B : Word; const Limit : Word )
5302: Procedure DeclimI( var B : Integer; const Limit : Integer )
5303: Procedure DeclimL( var B : LongInt; const Limit : LongInt )
5304: Function MaxB( const B1, B2 : Byte ) : Byte
5305: Function MinB( const B1, B2 : Byte ) : Byte
5306: Function MaxSI( const B1, B2 : ShortInt ) : ShortInt
5307: Function MinSI( const B1, B2 : ShortInt ) : ShortInt
5308: Function MaxW( const B1, B2 : Word ) : Word
5309: Function MinW( const B1, B2 : Word ) : Word
5310: Function esbMaxI( const B1, B2 : Integer ) : Integer
5311: Function esbMinI( const B1, B2 : Integer ) : Integer
5312: Function MaxL( const B1, B2 : LongInt ) : LongInt
5313: Function MinL( const B1, B2 : LongInt ) : LongInt
5314: Procedure SwapB( var B1, B2 : Byte )
5315: Procedure SwapSI( var B1, B2 : ShortInt )
5316: Procedure SwapW( var B1, B2 : Word )

```

```

5317: Procedure SwapI( var B1, B2 : SmallInt)
5318: Procedure SwapL( var B1, B2 : LongInt)
5319: Procedure SwapI32( var B1, B2 : Integer)
5320: Procedure SwapC( var B1, B2 : LongWord)
5321: Procedure SwapInt64( var X, Y : Int64)
5322: Function esbSign( const B : LongInt) : ShortInt
5323: Function Max4Word( const X1, X2, X3, X4 : Word) : Word
5324: Function Min4Word( const X1, X2, X3, X4 : Word) : Word
5325: Function Max3Word( const X1, X2, X3 : Word) : Word
5326: Function Min3Word( const X1, X2, X3 : Word) : Word
5327: Function MaxBArray( const B : array of Byte) : Byte
5328: Function MaxWArray( const B : array of Word) : Word
5329: Function MaxSIArray( const B : array of ShortInt) : ShortInt
5330: Function MaxIArray( const B : array of Integer) : Integer
5331: Function MaxLArray( const B : array of LongInt) : LongInt
5332: Function MinBArray( const B : array of Byte) : Byte
5333: Function MinWArray( const B : array of Word) : Word
5334: Function MinSIArray( const B : array of ShortInt) : ShortInt
5335: Function MinIArray( const B : array of Integer) : Integer
5336: Function MinLArray( const B : array of LongInt) : LongInt
5337: Function SumBArray( const B : array of Byte) : Byte
5338: Function SumBArray2( const B : array of Byte) : Word
5339: Function SumSIArray( const B : array of ShortInt) : ShortInt
5340: Function SumSIArray2( const B : array of ShortInt) : Integer
5341: Function SumWArray( const B : array of Word) : Word
5342: Function SumWArray2( const B : array of Word) : LongInt
5343: Function SumIArray( const B : array of Integer) : Integer
5344: Function SumLArray( const B : array of LongInt) : LongInt
5345: Function SumLWArray( const B : array of LongWord) : LongWord
5346: Function ESBDigits( const X : LongWord) : Byte
5347: Function BitsHighest( const X : LongWord) : Integer
5348: Function ESBBitsNeeded( const X : LongWord) : Integer
5349: Function esbGCD( const X, Y : LongWord) : LongWord
5350: Function esbLCM( const X, Y : LongInt) : Int64
5351: //Function esbLCM( const X, Y : LongInt) : LongInt
5352: Function RelativePrime( const X, Y : LongWord) : Boolean
5353: Function Get87ControlWord : TBitList
5354: Procedure Set87ControlWord( const CWord : TBitList)
5355: Procedure SwapExt( var X, Y : Extended)
5356: Procedure SwapDbl( var X, Y : Double)
5357: Procedure SwapSing( var X, Y : Single)
5358: Function esbSgn( const X : Extended) : ShortInt
5359: Function Distance( const X1, Y1, X2, Y2 : Extended) : Extended
5360: Function ExtMod( const X, Y : Extended) : Extended
5361: Function ExtRem( const X, Y : Extended) : Extended
5362: Function CompMOD( const X, Y : Comp) : Comp
5363: Procedure Polar2XY( const Rho, Theta : Extended; var X, Y : Extended)
5364: Procedure XY2Polar( const X, Y : Extended; var Rho, Theta : Extended)
5365: Function DMS2Extended( const Degs, Mins, Secs : Extended) : Extended
5366: Procedure Extended2DMS( const X : Extended; var Degs, Mins, Secs : Extended)
5367: Function MaxExt( const X, Y : Extended) : Extended
5368: Function MinExt( const X, Y : Extended) : Extended
5369: Function MaxEArray( const B : array of Extended) : Extended
5370: Function MinEArray( const B : array of Extended) : Extended
5371: Function MaxSArray( const B : array of Single) : Single
5372: Function MinSArray( const B : array of Single) : Single
5373: Function MaxCArray( const B : array of Comp) : Comp
5374: Function MinCArray( const B : array of Comp) : Comp
5375: Function SumSArray( const B : array of Single) : Single
5376: Function SumEArray( const B : array of Extended) : Extended
5377: Function SumSqEArray( const B : array of Extended) : Extended
5378: Function SumSqDiffEArray( const B : array of Extended; Diff : Extended) : Extended
5379: Function SumXYEArray( const X, Y : array of Extended) : Extended
5380: Function SumCArray( const B : array of Comp) : Comp
5381: Function FactorialX( A : LongWord) : Extended
5382: Function PermutationX( N, R : LongWord) : Extended
5383: Function esbBinomialCoeff( N, R : LongWord) : Extended
5384: Function IsPositiveEArray( const X : array of Extended) : Boolean
5385: Function esbGeometricMean( const X : array of Extended) : Extended
5386: Function esbHarmonicMean( const X : array of Extended) : Extended
5387: Function ESBMean( const X : array of Extended) : Extended
5388: Function esbSampleVariance( const X : array of Extended) : Extended
5389: Function esbPopulationVariance( const X : array of Extended) : Extended
5390: Procedure esbSampleVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5391: Procedure esbPopulationVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5392: Function GetMedian( const SortedX : array of Extended) : Extended
5393: Function GetMode( const SortedX : array of Extended; var Mode : Extended) : Boolean
5394: Procedure GetQuartiles( const SortedX : array of Extended; var Q1, Q3 : Extended)
5395: Function ESBMagnitude( const X : Extended) : Integer
5396: Function ESBTan( Angle : Extended) : Extended
5397: Function ESB Cot( Angle : Extended) : Extended
5398: Function ESB Cosec( const Angle : Extended) : Extended
5399: Function ESB Sec( const Angle : Extended) : Extended
5400: Function ESB ArcTan( X, Y : Extended) : Extended
5401: Procedure ESB SinCos( Angle : Extended; var SinX, CosX : Extended)
5402: Function ESB ArcCos( const X : Extended) : Extended
5403: Function ESB ArcSin( const X : Extended) : Extended
5404: Function ESB ArcSec( const X : Extended) : Extended
5405: Function ESB ArcCosec( const X : Extended) : Extended

```

```

5406: Function ESBLog10( const X : Extended ) : Extended
5407: Function ESBLog2( const X : Extended ) : Extended
5408: Function ESBLogBase( const X, Base : Extended ) : Extended
5409: Function Pow2( const X : Extended ) : Extended
5410: Function IntPow( const Base : Extended; const Exponent : LongWord ) : Extended
5411: Function ESBIntPower( const X : Extended; const N : LongInt ) : Extended
5412: Function XtoY( const X, Y : Extended ) : Extended
5413: Function esbTenToY( const Y : Extended ) : Extended
5414: Function esbTwoToY( const Y : Extended ) : Extended
5415: Function LogXtoBaseY( const X, Y : Extended ) : Extended
5416: Function esbISqrt( const I : LongWord ) : Longword
5417: Function ILG2( const I : LongWord ) : Longword
5418: Function IGreatestPowerOf2( const N : LongWord ) : LongWord
5419: Function ESBArCosh( X : Extended ) : Extended
5420: Function ESBArSinh( X : Extended ) : Extended
5421: Function ESBArTanh( X : Extended ) : Extended
5422: Function ESBCosinh( X : Extended ) : Extended
5423: Function ESBSinh( X : Extended ) : Extended
5424: Function ESBTanh( X : Extended ) : Extended
5425: Function InverseGamma( const X : Extended ) : Extended
5426: Function esbGamma( const X : Extended ) : Extended
5427: Function esbLnGamma( const X : Extended ) : Extended
5428: Function esbBeta( const X, Y : Extended ) : Extended
5429: Function IncompleteBeta( X : Extended; P, Q : Extended ) : Extended
5430: end;
5431:
5432: ***** Integer Huge Cardinal Utils*****
5433: Function Add_uint64_WithCarry( x, y : uint64; var Carry : Boolean ) : uint64
5434: Function Add_uint32_WithCarry( x, y : uint32; var Carry : Boolean ) : uint32
5435: Function Subtract_uint64_WithBorrow( x, y : uint64; var Borrow : Boolean ) : uint64
5436: Function Subtract_uint32_WithBorrow( x, y : uint32; var Borrow : Boolean ) : uint32
5437: Function BitCount_8( Value : byte ) : integer
5438: Function BitCount_16( Value : uint16 ) : integer
5439: Function BitCount_32( Value : uint32 ) : integer
5440: Function BitCount_64( Value : uint64 ) : integer
5441: Function CountSetBits_64( Value : uint64 ) : integer TPrimalityTestNoticeProc',
5442: Procedure ( CountPrimalityTests : integer )
5443: Function gcd( a, b : THugeCardinal ) : THugeCardinal
5444: Function lcm( a, b : THugeCardinal ) : THugeCardinal
5445: Function isCoPrime( a, b : THugeCardinal ) : boolean
5446: Function isProbablyPrime( p: THugeCardinal; OnProgress: TProgress; var wasAborted: boolean): boolean
5447: Function hasSmallFactor( p : THugeCardinal ) : boolean
5448: //Function GeneratePrime( NumBits : integer; OnProgress : TProgress; OnPrimalityTest:
      TPrimalityTestNoticeProc; PassCount: integer; Pool1:TMemoryStreamPool; var Prime: THugeCardinal; var
      NumbersTested: integer ) : boolean
5449: Function Inverse( Prime, Modulus : THugeCardinal ) : THugeCardinal
5450: Const('StandardExponent','LongInt'( 65537 );
5451: //Procedure Compute_RSA_Fundamentals_2Factors(RequiredBitLengthOfN:integer;Fixed_e:uint64;var N,e,d,
      Totient:TProgress;OnPrimalityTest:TPrimalityTestNoticeProc;GeneratePrimePassCount:int;Pool1:TMemoryStreamPo
      Numbers
5452: Function Validate_RSA_Fundamentals( var N, e, d, Totient : THugeCardinal ) : boolean')
5453:
5454: procedure SIRegister_xrtl_math_Integer(CL: TPSPascalCompiler);
5455: begin
5456:   AddTypeS('TXRTLInteger', 'array of Integer'
5457:   AddClassN(FindClass('TOBJECT'), 'EXRTLMATHException'
5458:   (FindClass('TOBJECT'), 'EXRTLExtendInvalidArgument'
5459:   AddClassN(FindClass('TOBJECT'), 'EXRTLDivisionByZero'
5460:   AddClassN(FindClass('TOBJECT'), 'EXRTLExpInvalidArgument'
5461:   AddClassN(FindClass('TOBJECT'), 'EXRTLInvalidRadix'
5462:   AddClassN(FindClass('TOBJECT'), 'EXRTLInvalidRadixDigit'
5463:   AddClassN(FindClass('TOBJECT'), 'EXRTLRootInvalidArgument'
5464:   'BitsPerByte','LongInt'( 8 );
5465:   BitsPerDigit','LongInt'( 32 );
5466:   SignBitMask','LongWord( $80000000 );
5467:   Function XRTLAdjustBits( const ABits : Integer ) : Integer
5468:   Function XRTLLength( const AInteger : TXRTLInteger ) : Integer
5469:   Function XRTLDataBits( const AInteger : TXRTLInteger ) : Integer
5470:   Procedure XRTLBBitPosition( const BitIndex : Integer; var Index, Mask : Integer )
5471:   Procedure XRTLBBitSet( var AInteger : TXRTLInteger; const BitIndex : Integer )
5472:   Procedure XRTLBBitReset( var AInteger : TXRTLInteger; const BitIndex : Integer )
5473:   Function XRTLBGet( const AInteger : TXRTLInteger; const BitIndex : Integer ) : Integer
5474:   Function XRTLBGetBool( const AInteger : TXRTLInteger; const BitIndex : Integer ) : Boolean
5475:   Function XRTLExtend(const AInteger:TXRTLInteger;ADataBits:Integer;Sign:Int8;var AResult:TXRTLInteger):Int;
5476:   Function XRTLZeroExtend(const AInteger:TXRTLInteger;ADataBits:Integer; var AResult:TXRTLInteger):Integer;
5477:   Function XRTLSignExtend(const AInteger:TXRTLInteger; ADataBits:Integer;var AResult:TXRTLInteger):Integer;
5478:   Function XRTLSignStrip(const AInteger:TXRTLInteger;var AResult:TXRTLInteger;const AMinDataBits:Int):Int;
5479:   Procedure XRTLNNot( const AInteger : TXRTLInteger; var AResult : TXRTLInteger )
5480:   Procedure XRTLOr( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger )
5481:   Procedure XRTLAnd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger )
5482:   Procedure XRTLXor( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger )
5483:   Function XRTLSign( const AInteger : TXRTLInteger ) : Integer
5484:   Procedure XRTLZero( var AInteger : TXRTLInteger )
5485:   Procedure XRTLOne( var AInteger : TXRTLInteger )
5486:   Procedure XRTLMOne( var AInteger : TXRTLInteger )
5487:   Procedure XRTLTTwo( var AInteger : TXRTLInteger )
5488:   Function XRTLNeg( const AInteger : TXRTLInteger; var AResult : TXRTLInteger ) : Integer
5489:   Function XRTLAbs( const AInteger : TXRTLInteger; var AResult : TXRTLInteger ) : Integer
5490:   Procedure XRTLFULLSum( const A, B, C : Integer; var Sum, Carry : Integer )

```

```

5491: Function XRTLAdd( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger ) : Integer;
5492: Function XRTLAdd1(const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5493: Function XRTLSUB( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger ) : Integer;
5494: Function XRTLSUB1( const AInteger1:TXRTLInteger;const AInteger2:Int64;var AResult:TXRTLInteger):Integer;
5495: Function XRTLCompare( const AInteger1, AInteger2 : TXRTLInteger ) : Integer;
5496: Function XRTLCompare1( const AInteger1 : TXRTLInteger; const AInteger2 : Int64 ) : Integer;
5497: Function XRTLUMUL( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger ) : Integer;
5498: Function XRTLMULAdd(const AInteger1,AInteger2,TXRTLInteger; var AResult:TXRTLInteger):Integer;
5499: Function XRTLMUL( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger ) : Integer;
5500: Procedure XRTLDIVMod( const AInteger1, AInteger2 : TXRTLInteger; var QResult, RResult : TXRTLInteger );
5501: Procedure XRTLSQR( const AInteger : TXRTLInteger; var AResult : TXRTLInteger );
5502: Procedure XRTLSQR1( const AInteger : TXRTLInteger; var AResult : TXRTLInteger );
5503: Procedure XRTLRoot( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger );
5504: Procedure XRTLRootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
AHIGHApproxResult:TXRTLInteger);
5505: Procedure XRTLRootApprox(const AInteger1,AInteger2:TXRTLInteger;var ALowApproxResult,
AHIGHApproxResult:TXRTLInteger);
5506: Procedure XRTLEXP( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger );
5507: Procedure XRTLEXPMOD( const AInteger1, AInteger2, AInteger3 : TXRTLInteger;var AResult:TXRTLInteger );
5508: Procedure XRTLSLBL( const AInteger : TXRTLInteger; const BitCount:Integer; var AResult : TXRTLInteger );
5509: Procedure XRTLSABL( const AInteger : TXRTLInteger; const BitCount:Integer; var AResult : TXRTLInteger );
5510: Procedure XRTLRCBL( const AInteger : TXRTLInteger; const BitCount:Integer; var AResult : TXRTLInteger );
5511: Procedure XRTLSLDL( const AInteger:TXRTLInteger;const DigitCount:Integer; var AResult:TXRTLInteger );
5512: Procedure XRTLSADL( const AInteger : TXRTLInteger; const DigitCount:Integer;var AResult:TXRTLInteger );
5513: Procedure XRTLRCDL( const AInteger:TXRTLInteger;const DigitCount:Integer; var AResult:TXRTLInteger );
5514: Procedure XRTLSLBR( const AInteger : TXRTLInteger; const BitCount:Integer; var AResult : TXRTLInteger );
5515: Procedure XRTLSABR( const AInteger : TXRTLInteger; const BitCount:Integer; var AResult : TXRTLInteger );
5516: Procedure XRTLRCBR( const AInteger : TXRTLInteger; const BitCount:Integer; var AResult : TXRTLInteger );
5517: Procedure XRTLSLDR( const AInteger:TXRTLInteger; const DigitCount:Integer; var AResult:TXRTLInteger );
5518: Procedure XRTLSADR( const AInteger : TXRTLInteger; const DigitCount:Integer;var AResult:TXRTLInteger );
5519: Procedure XRTLRCDR( const AInteger : TXRTLInteger;const DigitCount:Integer;var AResult:TXRTLInteger );
5520: Function XRTLToHex( const AInteger : TXRTLInteger; Digits : Integer ) : string;
5521: Function XRTLToBin( const AInteger : TXRTLInteger; Digits : Integer ) : string;
5522: Function XRTLToString( const AInteger : TXRTLInteger; Radix : Integer; Digits : Integer ) : string;
5523: Procedure XRTLFROMHex( const Value : string; var AResult : TXRTLInteger );
5524: Procedure XRTLFROMBin( const Value : string; var AResult : TXRTLInteger );
5525: Procedure XRTLFROMString( const Value : string; var AResult : TXRTLInteger; Radix : Integer );
5526: Procedure XRTLASSIGN( const AInteger : TXRTLInteger; var AResult : TXRTLInteger );
5527: Procedure XRTLASSIGN1( const Value : Integer; var AResult : TXRTLInteger );
5528: Procedure XRTLASSIGN2( const Value : Int64; var AResult : TXRTLInteger );
5529: Procedure XRTLAPPEND( const ALow, AHIGH : TXRTLInteger; var AResult : TXRTLInteger );
5530: Procedure XRTLSPLIT( const AInteger : TXRTLInteger; var ALow,AHIGH : TXRTLInteger;LowDigits: Integer );
5531: Function XRTLGetMSBIndex( const AInteger : TXRTLInteger ) : Integer;
5532: Procedure XRTLMINMAX( const AInteger1, AInteger2 : TXRTLInteger;var AMinResult,AMaxResult : TXRTLInteger );
5533: Procedure XRTLMIN( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger );
5534: Procedure XRTLMIN1( const AInteger1:TXRTLInteger;const AInteger2:Integer;var AResult:TXRTLInteger );
5535: Procedure XRTLMAX( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger );
5536: Procedure XRTLMAX1( const AInteger1:TXRTLInteger;const AInteger2:Integer;var AResult:TXRTLInteger );
5537: Procedure XRTLGCDF( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger );
5538: Procedure XRTLSWAP( var AInteger1, AInteger2 : TXRTLInteger );
5539: Procedure XRTLFACCTORIAL( const AInteger : TXRTLInteger; var AResult : TXRTLInteger );
5540: Procedure XRTLFACCTORIALMOD( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger );
5541: end;
5542:
5543: procedure SIRegister_JvXPCoreUtils(CL: TPSCompiler);
5544: begin
5545:   Function JvXPMethodsEqual( const Method1, Method2 : TMethod ) : Boolean;
5546:   Procedure JvXPDrawLine( const ACanvas : TCanvas; const X1, Y1, X2, Y2 : Integer );
5547:   Procedure JvXPCreateGradientRect( const AWidth, AHeight : Integer; const StartColor, EndColor : TColor;
      const Colors:TJvXPGradientColors;const Style:TJvXPGradientStyle;const Dithered:Boolean;var Bitmap:TBitmap );
5548:   Procedure JvXPADJUSTBORDRECT( const BorderWidth:Byte; const ShowBoundLines:Boolean; const
      BoundLines:TJvXPBoundLines; var Rect : TRect );
5549:   Procedure JvXPDRAWBOUNDLINES(const ACan:TCanvas;const BoundLns:TJvXPBoundLns;const
      AColor:TColor;Rect:TRect );
5550:   Procedure JvXPConvertToGray2( Bitmap : TBitmap );
5551:   Procedure JvXPRENDERTEXT( const AParent : TControl; const ACanvas : TCanvas; ACaption : TCaption; const
      AFont : TFont; const AEnabled, AShowAccelChar : Boolean; var ARect : TRect; AFlags : Integer );
5552:   Procedure JvXPFrame3D(const ACanvas:TCanvas;const Rect:TRect; TopColor,BottomColor:TColor;const
      Swapped:Bool ;
5553:   Procedure JvXPColorizeBitmap( Bitmap : TBitmap; const AColor : TColor );
5554:   Procedure JvXPSETDRAWFLAGS(const AAlignment: TALIGNMENT;const AWORDWRAP: Boolean; var Flags : Integer );
5555:   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 );
5556: end;
5557:
5558:
5559: procedure SIRegister_uwinstr(CL: TPSCompiler);
5560: begin
5561:   Function StrDec( S : String ) : String;
5562:   Function uIsNumeric( var S : String; var X : Float ) : Boolean;
5563:   Function ReadNumFromEdit( Edit : TEdit ) : Float;
5564:   Procedure WriteNumToFile( var F : Text; X : Float );
5565: end;
5566:
5567: procedure SIRegister_utexplor(CL: TPSCompiler);
5568: begin
5569:   Function TeX_InitGraphics( FileName : String; PgWidth, PgHeight : Integer; Header : Boolean ) : Boolean;
5570:   Procedure TeX_SetWindow( X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean );

```

```

5571: Procedure TeX_LeaveGraphics( Footer : Boolean)
5572: Procedure TeX_SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
5573: Procedure TeX_SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
5574: Procedure TeX_SetGraphTitle( Title : String)
5575: Procedure TeX_SetOxTitle( Title : String)
5576: Procedure TeX_SetOyTitle( Title : String)
5577: Procedure TeX_PlotOxAxis
5578: Procedure TeX_PlotOyAxis
5579: Procedure TeX_PlotGrid( Grid : TGrid)
5580: Procedure TeX_WriteGraphTitle
5581: Function TeX_SetMaxCurv( NCurv : Byte) : Boolean
5582: Procedure TeX_SetPointParam( CurvIndex, Symbol, Size : Integer)
5583: Procedure TeX_SetLineParam( CurvIndex, Style : Integer; Width : Float; Smooth : Boolean)
5584: Procedure TeX_SetCurvLegend( CurvIndex : Integer; Legend : String)
5585: Procedure TeX_SetCurvStep( CurvIndex, Step : Integer)
5586: Procedure TeX_PlotCurve( X, Y : TVector; Lb, Ub, CurvIndex : Integer)
5587: Procedure TeX_PlotCurveWithErrorBars( X, Y, S : TVector; Ns, Lb, Ub, CurvIndex : Integer)
5588: Procedure TeX_PlotFunc( Func : TFunc; X1, X2 : Float; Npt : Integer; CurvIndex : Integer)
5589: Procedure TeX_WriteLegend( NCurv : Integer; ShowPoints, ShowLines : Boolean)
5590: Procedure TeX_ConRec( Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
5591: Function Xcm( X : Float) : Float
5592: Function Ycm( Y : Float) : Float
5593: end;
5594:
5595: *-----*)
5596: procedure SIRegister_VarRecUtils(CL: TPSPascalCompiler);
5597: begin
5598:   TConstArray', 'array of TVarRec
5599:   Function CopyVarRec( const Item : TVarRec) : TVarRec
5600:   Function CreateConstArray( const Elements : array of const) : TConstArray
5601:   Procedure FinalizeVarRec( var Item : TVarRec)
5602:   Procedure FinalizeConstArray( var Arr : TConstArray)
5603: end;
5604:
5605: procedure SIRegister_StStrs(CL: TPSPascalCompiler);
5606: begin
5607:   Function HexBS( B : Byte) : ShortString
5608:   Function HexWS( W : Word) : ShortString
5609:   Function HexLS( L : LongInt) : ShortString
5610:   Function HexPtrS( P : Pointer) : ShortString
5611:   Function BinaryBS( B : Byte) : ShortString
5612:   Function BinaryWS( W : Word) : ShortString
5613:   Function BinaryLS( L : LongInt) : ShortString
5614:   Function OctalBS( B : Byte) : ShortString
5615:   Function OctalWS( W : Word) : ShortString
5616:   Function OctalLS( L : LongInt) : ShortString
5617:   Function Str2Int16S( const S : ShortString; var I : SmallInt) : Boolean
5618:   Function Str2WordS( const S : ShortString; var I : Word) : Boolean
5619:   Function Str2LongS( const S : ShortString; var I : LongInt) : Boolean
5620:   Function Str2Reals( const S : ShortString; var R : Double) : Boolean
5621:   Function Str2RealS( const S : ShortString; var R : Real) : Boolean
5622:   Function Str2ExtS( const S : ShortString; var R : Extended) : Boolean
5623:   Function Long2StrS( L : LongInt) : ShortString
5624:   Function Real2StrS( R : Double; Width : Byte; Places : ShortInt) : ShortString
5625:   Function Ext2StrS( R : Extended; Width : Byte; Places : ShortInt) : ShortString
5626:   Function ValPrepS( const S : ShortString) : ShortString
5627:   Function CharStrS( C : AnsiChar; Len : Cardinal) : ShortString
5628:   Function PadChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5629:   Function PadS( const S : ShortString; Len : Cardinal) : ShortString
5630:   Function LeftPadChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5631:   Function LeftPadS( const S : ShortString; Len : Cardinal) : ShortString
5632:   Function TrimLeadS( const S : ShortString) : ShortString
5633:   Function TrimTrailsS( const S : ShortString) : ShortString
5634:   Function TrimS( const S : ShortString) : ShortString
5635:   Function TrimSpacesS( const S : ShortString) : ShortString
5636:   Function CenterChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5637:   Function Centers( const S : ShortString; Len : Cardinal) : ShortString
5638:   Function EntabS( const S : ShortString; TabSize : Byte) : ShortString
5639:   Function DetabS( const S : ShortString; TabSize : Byte) : ShortString
5640:   Function ScrambleS( const S : ShortString) : ShortString
5641:   Function SubstituteS( const S, Key : ShortString) : ShortString
5642:   Function Filters( const S, Filters : ShortString) : ShortString
5643:   Function CharExistsS( const S : ShortString; C : AnsiChar) : Boolean
5644:   Function CharCounts( const S : ShortString; C : AnsiChar) : Byte
5645:   Function WordCountS( const S, WordDelims : ShortString) : Cardinal
5646:   Function WordPositionS( N : Cardinal; const S, WordDelims : ShortString; var Pos : Cardinal) : Boolean
5647:   Function ExtractWordS( N : Cardinal; const S, WordDelims : ShortString) : ShortString
5648:   Function AsciiCountS( const S, WordDelims : ShortString; Quote : AnsiChar) : Cardinal
5649:   Function AsciiPositionS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar;var Pos:Cardinal):Boolean
5650:   Function ExtractAsciiS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar): ShortString
5651:   Procedure WordWrapS(const InSt: ShortString; var OutSt,Overlap: ShortString;
Margin:Cardinal;PadToMargin:Boolean)
5652:   Function CompStringS( const S1, S2 : ShortString) : Integer
5653:   Function CompUCStringS( const S1, S2 : ShortString) : Integer
5654:   Function SoundexS( const S : ShortString) : ShortString
5655:   Function MakeLetterSetS( const S : ShortString) : Longint
5656:   Procedure BMMakeTableS( const MatchString : shortString; var BT : BTable)
5657:   Function BMSearchS(var Buffer,BufLength:Card;var BT:BTable;const MatchString:ShortString;var
Pos:Cardinal):Bool;

```

```

5658: Function BMSearchUCS(var Buffer,BufLength:Card;var BT:BTable;const MatchStr:ShortString;var
5659: Pos:Cardinal):Bool;
5660: Function DefaultExtensions( const Name, Ext : ShortString) : ShortString
5661: Function ForceExtensionS( const Name, Ext : shortString) : ShortString
5662: Function JustFilenameS( const PathName : ShortString) : ShortString
5663: Function JustNameS( const PathName : ShortString) : ShortString
5664: Function JustPathnameS( const PathName : ShortString) : ShortString
5665: Function AddBackSlashS( const DirName : ShortString) : ShortString
5666: Function CleanPathNameS( const PathName : ShortString) : ShortString
5667: Function HasExtensionS( const Name : ShortString; var DotPos : Cardinal) : Boolean
5668: Function CommaizeChs( L : LongInt) : ShortString
5669: Function CommaizeChs( L : Longint; Ch : AnsiChar) : ShortString
5670: Function FloatFormS(const Mask:ShortString;R:TstFloat;const LtCurr,RtCurr:SString;Sep,
DecPt:Char):ShortString;
5671: Function LongIntFormS(const Mask:ShortString;L:LongInt;const LtCurr,
RtCurr:ShortString;Sep:AnsiChar):ShortString;
5672: Function StrChPosS( const P : ShortString; C : AnsiChar; var Pos : Cardinal) : Boolean
5673: Function StrnposS( const P, S : ShortString; var Pos : Cardinal) : Boolean
5674: Function StrStCopyS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5675: Function StrChInsertS( const S : ShortString; C : AnsiChar; Pos : Cardinal) : ShortString
5676: Function StrStInsertS( const S1, S2 : ShortString; Pos : Cardinal) : ShortString
5677: Function StrChDeleteS( const S : ShortString; Pos : Cardinal) : ShortString
5678: Function StrStDeleteS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5679: Function ContainsOnlyS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5680: Function ContainsOtherThanS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5681: Function CopyLeftS( const S : ShortString; Len : Cardinal) : ShortString
5682: Function CopyMidS( const S : ShortString; First, Len : Cardinal) : ShortString
5683: Function CopyRights( const S : ShortString; First : Cardinal) : shortString
5684: Function CopyRightAbsS( const S : ShortString; NumChars : Cardinal) : ShortString
5685: Function CopyFromNthWordS(const S,WordDelims:string;const AWord:String;N:Card;var
SubString:ShortString):Bool;
5686: Function DeleteFromNthWordS(const S,WordDelims:String;AWord:ShortString;N:Card;var
SubStr:ShortString):Bool;
5687: Function CopyFromToWordsS(const S,WordDelims,Word1,Word2:ShortString;N1,N2:Card;var
SubString:ShortString):Bool;
5688: Function DeleteFromToWordsS(const S,WordDelims,Wrd1,Wrd2:ShortString;N1,N2:Card;var
SubString:ShortString):Bool;
5689: Function CopyWithinS( const S, Delimiter : ShortString; Strip : Boolean) : ShortString
5690: Function DeleteWithinS( const S, Delimiter : ShortString) : ShortString
5691: Function ExtractTokensS(const S,
Delims:ShortString;QuoteChar:AnsiChar;AllowNulls:Boolean;Tokens:TStrings):Cardinal
5692: Function IsChAlphas( C : Char) : Boolean
5693: Function IsChNumericS( C : Char; const Numbers : ShortString) : Boolean
5694: Function IsChAlphaNumerics( C : Char; const Numbers : shortString) : Boolean
5695: Function IsStrAlphaS( const S : Shortstring) : Boolean
5696: Function IsStrNumericS( const S, Numbers : ShortString) : Boolean
5697: Function IsStrAlphaNumericS( const S, Numbers : ShortString) : Boolean
5698: Function LastWordS( const S, WordDelims, AWord : ShortString; var Position : Cardinal) : Boolean
5699: Function LastWordAbsS( const S, WordDelims : ShortString; var Position : Cardinal) : Boolean
5700: Function LastStringS( const S, AString : ShortString; var Position : Cardinal) : Boolean
5701: Function LeftTrimCharsS( const S, Chars : ShortString) : ShortString
5702: Function KeepCharsS( const S, Chars : ShortString) : ShortString
5703: Function RepeatStringS(const RepeatString:ShortString;var Repetitions: Cardinal; MaxLen :
Cardinal):ShortString;
5704: Function ReplaceStringS(const S,OldStr,NewStr:ShortString;N:Cardinal;var
Replacements:Cardinal):ShortString;
5705: Function ReplaceStringAllS(const S,OldString,NewString:ShortString;var Replacements:Cardinal):ShortString;
5706: Function ReplaceWordsS(const S,WordDelims,OldWord,NewW:SString;N:Cardinal;var
Replacements:Cardinal):ShortString
5707: Function ReplaceWordAllS(const S,WordDelims,OldWord,NewWord:ShortString;var
Replacements:Cardinal):ShortString
5708: Function RightTrimCharsS( const S, Chars : ShortString) : ShortString
5709: Function StrWithinS(const S,SearchStr: ShortString;Start:Cardinal;var Position:Cardinal):boolean
5710: Function TrimCharsS( const S, Chars : ShortString) : ShortString
5711: Function WordPosS(const S,WordDelims,AWord:ShortString;N:Cardinal; var Position: Cardinal):Boolean
5712: end;
5713;
5714;
5715: *****unit uPSI_StUtils; from Systools4*****
5716: Function SignL( L : LongInt) : Integer
5717: Function SignF( F : Extended) : Integer
5718: Function MinWord( A, B : Word) : Word
5719: Function MidWord( W1, W2, W3 : Word) : Word
5720: Function MaxWord( A, B : Word) : Word
5721: Function MinLong( A, B : LongInt) : LongInt
5722: Function MidLong( L1, L2, L3 : LongInt) : LongInt
5723: Function MaxLong( A, B : LongInt) : LongInt
5724: Function MinFloat( F1, F2 : Extended) : Extended
5725: Function MidFloat( F1, F2, F3 : Extended) : Extended
5726: Function MaxFloat( F1, F2 : Extended) : Extended
5727: Function MakeInteger16( H, L : Byte) : SmallInt
5728: Function MakeWordS( H, L : Byte) : Word
5729: Function SwapNibble( B : Byte) : Byte
5730: Function SwapWord( L : LongInt) : LongInt
5731: Procedure SetFlag( var Flags : Word; FlagMask : Word)
5732: Procedure ClearFlag( var Flags : Word; FlagMask : Word)
5733: Function FlagIsSet( Flags, FlagMask : Word) : Boolean
5734: Procedure SetByteFlag( var Flags : Byte; FlagMask : Byte)

```

```

5735: Procedure ClearByteFlag( var Flags : Byte; FlagMask : Byte)
5736: Function ByteFlagIsSet( Flags, FlagMask : Byte) : Boolean
5737: Procedure SetLongFlag( var Flags : LongInt; FlagMask : LongInt)
5738: Procedure ClearLongFlag( var Flags : LongInt; FlagMask : LongInt)
5739: Function LongFlagIsSet( Flags, FlagMask : LongInt) : Boolean
5740: Procedure ExchangeBytes( var I, J : Byte)
5741: Procedure ExchangeWords( var I, J : Word)
5742: Procedure ExchangeLongInts( var I, J : LongInt)
5743: Procedure ExchangeStructs( var I, J, Size : Cardinal)
5744: Procedure FillWord( var Dest, Count : Cardinal; Filler : Word)
5745: Procedure FillStruct( var Dest, Count : Cardinal; var Filler, FillerSize : Cardinal)
5746: Function AddWordToPtr( P : __Pointer; W : Word) : __Pointer
5747: //*****uPSI_STFIN*****
5748: Function AccruedInterestMaturity(Issue,Maturity:TStDate;Rate,Par:Extended;Basis: TStBasis) : Extended
5749: Function AccruedInterestPeriodic(Issue,Settlement,Maturity:TStDate;Rate,
      Par:Extended;Frequency:TStFrequency; Basis : TStBasis) : Extended
5750: Function BondDuration( Settlement,Maturity:TStDate;Rate,
      Yield:Ext;Frequency:TStFrequency;Basis:TStBasis):Extended
5751: Function BondPrice(Settlement,Maturity:TStDate;Rate,Yield,Redempt:Ext;Freq:TStFrequency;Basis:TStBasis):
      Extended
5752: Function CumulativeInterest( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
      Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended
5753: Function CumulativePrincipal( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
      Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended
5754: Function DayCount( Day1, Day2 : TStDate; Basis : TStBasis) : LongInt
5755: Function DecliningBalance( Cost, Salvage : Extended; Life, Period, Month : Integer) : Extended
5756: Function DiscountRate(Settlement,Maturity:TStDate; Price,Redemption:Extended;Basis:TStBasis) : Extended
5757: Function DollarToDecimal( FracDollar : Extended; Fraction : Integer) : Extended
5758: Function DollarToDecimalText( DecDollar : Extended) : string
5759: Function DollarToFraction( DecDollar : Extended; Fraction : Integer) : Extended
5760: Function DollarToFractionStr( FracDollar : Extended; Fraction : Integer) : string
5761: Function EffectiveInterestRate( NominalRate : Extended; Frequency : TStFrequency) : Extended
5762: Function FutureValueS(Rate:Extended;NPeriods:Int;Pmt,
      PV:Extended;Freq:TStFreq;Timing:TStPaymentTime):Extended
5763: Function FutureValueSchedule( Principal : Extended; const Schedule : array of Double) : Extended
5764: Function FutureValueSchedule16( Principal : Extended; const Schedule, NRates : Integer) : Extended
5765: Function InterestRateS(NPeriods:Int;Pmt,PV,
      FV:Extended;Freq:TStFrequency;Timing:TStPaymentTime;Guess:Extended):Extend
5766: Function InternalRateOfReturn( const Values : array of Double; Guess : Extended) : Extended
5767: Function InternalRateOfReturn16( const Values, NValues : Integer; Guess : Extended) : Extended
5768: Function IsCardValid( const S : string) : Boolean
5769: Function ModifiedDuration(Settlement,Maturity:TStDate;Rate,
      Yield:Extended;Freq:TStFrequency;Basis:TStBasis):Extended
5770: Function ModifiedIRR(const Values : array of Double; FinanceRate,ReinvestRate: Extended) : Extended
5771: Function ModifiedIRR16(const Values,NValues:Integer; FinanceRate,ReinvestRate: Extended) : Extended
5772: Function NetPresentValueS( Rate : Extended; const Values : array of Double) : Extended
5773: Function NetPresentValue16( Rate : Extended; const Values, NValues : Integer) : Extended
5774: Function NominalInterestRate( EffectRate : Extended; Frequency : TStFrequency) : Extended
5775: Function NonperiodicIRR(const Values:array of Double;const Dates:array of TStDate;Guess:Extended):Extended;
5776: Function NonperiodicNPV(Rate:Extended;const Values: array of Double;const Dates:array of TStDate):Extended;
5777: Function Payment( Rate : Extended; NPeriods : Integer; PV, FV : Extended; Frequency : TStFrequency; Timing
      : TStPaymentTime) : Extended
5778: Function Periods( Rate:Extended;Pmt,PV,FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Integer;
5779: Function PresentValueS( Rate : Extended; NPeriods: Integer; Pmt, FV: Extended; Frequency : TStFrequency;
      Timing : TStPaymentTime) : Extended
5780: Function ReceivedAtMaturity(Settlement,Maturity:TStDate;Invest,Discount:Extended;Basis:TStBasis):Extended;
5781: Function RoundToDecimal( Value : Extended; Places : Integer; Bankers : Boolean) : Extended
5782: Function TBillEquivYield( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5783: Function TBillPrice( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5784: Function TBillyield( Settlement, Maturity : TStDate; Price : Extended) : Extended
5785: Function VariableDecliningBalance( Cost, Salvage : Extended; Life : Integer; StartPeriod, EndPeriod,
      Factor : Extended; NoSwitch : boolean) : Extended
5786: Function YieldDiscounted(Settlement,Maturity:TStDate;Price,Redemption:Extended;Basis:TStBasis):Extended;
5787: Function YieldPeriodic(Settlement,Maturity:TStDate;Rate,Price,
      Redemption:Extended;Freq:TStFrequency;Basis:TStBasis):Extended
5788: Function YieldMaturity(Issue,Settlement,Maturity:TStDate;Rate,Price:Extended;Basis:TStBasis):Extended;
5789:
5790: //*****unit uPSI_StAstroP;
5791: Procedure PlanetsPos( JD : Double; var PA : TStPlanetsArray)
5792: //*****unit unit uPSI_StStat; Statistic Package of SysTools*****
5793: Function AveDev( const Data : array of Double) : Double
5794: Function AveDev16( const Data, NData : Integer) : Double
5795: Function Confidence( Alpha, StandardDev : Double; Size : LongInt) : Double
5796: Function Correlation( const Data1, Data2 : array of Double) : Double
5797: Function Correlation16( const Data1, Data2, NData : Integer) : Double
5798: Function Covariance( const Data1, Data2 : array of Double) : Double
5799: Function Covariance16( const Data1, Data2, NData : Integer) : Double
5800: Function DevSq( const Data : array of Double) : Double
5801: Function DevSql16( const Data, NData : Integer) : Double
5802: Procedure Frequency(const Data:array of Double;const Bins:array of Double;var Counts: array of LongInt);
5803: //Procedure Frequency16( const Data, NData : Integer; const Bins, NBins : Integer; var Counts)
5804: Function GeometricMeanS( const Data : array of Double) : Double
5805: Function GeometricMean16( const Data, NData : Integer) : Double
5806: Function HarmonicMeanS( const Data : array of Double) : Double
5807: Function HarmonicMean16( const Data, NData : Integer) : Double
5808: Function Largest( const Data : array of Double; K : Integer) : Double
5809: Function Largest16( const Data, NData : Integer; K : Integer) : Double
5810: Function MedianS( const Data : array of Double) : Double
5811: Function Median16( const Data, NData : Integer) : Double

```

```

5812: Function Mode( const Data : array of Double) : Double
5813: Function Mode16( const Data, NData : Integer) : Double
5814: Function Percentile( const Data : array of Double; K : Double) : Double
5815: Function Percentile16( const Data, NData : Integer; K : Double) : Double
5816: Function PercentRank( const Data : array of Double; X : Double) : Double
5817: Function PercentRank16( const Data, NData : Integer; X : Double) : Double
5818: Function Permutations( Number, NumberChosen : Integer) : Extended
5819: Function Combinations( Number, NumberChosen : Integer) : Extended
5820: Function Factorials( N : Integer) : Extended
5821: Function Rank( Number : Double; const Data : array of Double; Ascending : Boolean) : Integer
5822: Function Rank16( Number : Double; const Data, NData : Integer; Ascending : Boolean) : Integer
5823: Function Smallest( const Data : array of Double; K : Integer) : Double
5824: Function Smallest16( const Data, NData : Integer; K : Integer) : Double
5825: Function TrimMean( const Data : array of Double; Percent : Double) : Double
5826: Function TrimMean16( const Data, NData : Integer; Percent : Double) : Double
5827: AddTypes('TStLinEst', 'record B0 : Double; B1 : double; seB0 : double; seB1 : double; R2 : Double; sigma : Double; SSr : double; SSE : Double; F0 : Double; df : Integer; end')
5828: + '1 : Double; LF : TStLinEst; ErrorStats:Bool';
5829: Procedure LinEst(const KnownY:array of Double;const KnownX:array of Double;var LF:TStLinEst;ErrorStats:Bool);
5830: Procedure LogEst(const KnownY:array of Double;const KnownX:array of Double;var LF:TStLinEst;ErrorStats:Bool);
5831: Function Forecast(X: Double; const KnownY: array of Double; const KnownX: array of Double) : Double
5832: Function ForecastExponential(X:Double;const KnownY:array of Double;const KnownX:array of Double):Double
5833: Function Intercept( const KnownY : array of Double; const KnownX : array of Double) : Double
5834: Function RSquared( const KnownY : array of Double; const KnownX : array of Double) : Double
5835: Function Slope( const KnownY : array of Double; const KnownX : array of Double) : Double
5836: Function StandardErrorY( const KnownY : array of Double; const KnownX : array of Double) : Double
5837: Function BetaDist( X, Alpha, Beta, A, B : Single) : Single
5838: Function BetaInv( Probability, Alpha, Beta, A, B : Single) : Single
5839: Function BinomDist( Numbers, Trials : Integer; ProbabilityS : Single; Cumulative : Boolean) : Single
5840: Function CritBinom( Trials : Integer; ProbabilityS, Alpha : Single) : Integer
5841: Function ChiDist( X : Single; DegreesFreedom : Integer) : Single
5842: Function ChiInv( Probability : Single; DegreesFreedom : Integer) : Single
5843: Function ExponDist( X, Lambda : Single; Cumulative : Boolean) : Single
5844: Function FDist( X : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5845: Function FInv( Probability : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5846: Function LogNormDist( X, Mean, StandardDev : Single) : Single
5847: Function LogInv( Probability, Mean, StandardDev : Single) : Single
5848: Function NormDist( X, Mean, StandardDev : Single; Cumulative : Boolean) : Single
5849: Function NormInv( Probability, Mean, StandardDev : Single) : Single
5850: Function NormSDist( Z : Single) : Single
5851: Function NormSInv( Probability : Single) : Single
5852: Function Poisson( X : Integer; Mean : Single; Cumulative : Boolean) : Single
5853: Function TDist( X : Single; DegreesFreedom : Integer; TwoTails : Boolean) : Single
5854: Function TInv( Probability : Single; DegreesFreedom : Integer) : Single
5855: Function Erfc( X : Single) : Single
5856: Function GammaLn( X : Single) : Single
5857: Function LargestSort( const Data : array of Double; K : Integer) : Double
5858: Function SmallestSort( const Data : array of double; K : Integer) : Double
5859:
5860: procedure SIRegister_TSTSorter(CL: TPSPascalCompiler);
5861: Function OptimumHeapToUse( RecLen : Cardinal; NumRecs : LongInt) : LongInt
5862: Function MinimumHeapToUse( RecLen : Cardinal) : LongInt
5863: Function MergeInfo( MaxHeap : LongInt; RecLen : Cardinal; NumRecs : LongInt) : TMergeInfo
5864: Function DefaultMergeName( MergeNum : Integer) : string
5865: Procedure ArraySort( var A, RecLen, NumRecs : Cardinal; Compare : TUntypedCompareFunc)
5866:
5867: procedure SIRegister_StAstro(CL: TPSPascalCompiler);
5868: Function AmountOfSunlight( LD : TStDate; Longitude, Latitude : Double) : TStTime
5869: Function FixedRiseSet( LD : TStDate; RA, DC, Longitude, Latitude : Double) : TStRiseSetRec
5870: Function SunPos( UT : TStDateTimeRec) : TStPosRec
5871: Function SunPosPrim( UT : TStDateTimeRec) : TStSunXYZRec
5872: Function SunRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5873: Function Twilight( LD : TStDate; Longitude, Latitude : Double; TwiType:TStTwilight):TStRiseSetRec
5874: Function LunarPhase( UT : TStDateTimeRec) : Double
5875: Function MoonPos( UT : TStDateTimeRec) : TStMoonPosRec
5876: Function MoonRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5877: Function FirstQuarter( D : TStDate) : TStLunarRecord
5878: Function FullMoon( D : TStDate) : TStLunarRecord
5879: Function LastQuarter( D : TStDate) : TStLunarRecord
5880: Function NewMoon( D : TStDate) : TStLunarRecord
5881: Function NextFirstQuarter( D : TStDate) : TStDateTimeRec
5882: Function NextFullMoon( D : TStDate) : TStDateTimeRec
5883: Function NextLastQuarter( D : TStDate) : TStDateTimeRec
5884: Function NextNewMoon( D : TStDate) : TStDateTimeRec
5885: Function PrevFirstQuarter( D : TStDate) : TStDateTimeRec
5886: Function PrevFullMoon( D : TStDate) : TStDateTimeRec
5887: Function PrevLastQuarter( D : TStDate) : TStDateTimeRec
5888: Function PrevNewMoon( D : TStDate) : TStDateTimeRec
5889: Function SiderealTime( UT : TStDateTimeRec) : Double
5890: Function Solstice( Y, Epoch : Integer; Summer : Boolean) : TStDateTimeRec
5891: Function Equinox( Y, Epoch : Integer; Vernal : Boolean) : TStDateTimeRec
5892: Function SEaster( Y, Epoch : Integer) : TStDate
5893: Function DateToAjd( D : TDate) : Double
5894: Function HoursMin( RA : Double) : ShortString
5895: Function DegrMin( DC : Double) : ShortString
5896: Function AJDToDate( D : Double) : TDate
5897:
5898: Procedure SIRegister_StDate(CL: TPSPascalCompiler);

```

```

5899: Function CurrentDate : TStDate
5900: Function StValidDate( Day, Month, Year, Epoch : Integer ) : Boolean
5901: Function DMYtoStDate( Day, Month, Year, Epoch : Integer ) : TStDate
5902: Procedure StDateToDMY( Julian : TStDate; var Day, Month, Year : Integer )
5903: Function StIncDate( Julian : TStDate; Days, Months, Years : Integer ) : TStDate
5904: Function IncDateTrunc( Julian : TStDate; Months, Years : Integer ) : TStDate
5905: Procedure StDateDiff( Date1, Date2 : TStDate; var Days, Months, Years : Integer )
5906: Function BondDateDiff( Date1, Date2 : TStDate; DayBasis : TStBondDateType ) : TStDate
5907: Function WeekOfYear( Julian : TStDate ) : Byte
5908: Function AstJulianDate( Julian : TStDate ) : Double
5909: Function AstJulianDateToStDate( AstJulian : Double; Truncate : Boolean ) : TStDate
5910: Function AstJulianDatePrim( Year, Month, Date : Integer; UT : TStTime ) : Double
5911: Function StDayOfWeek( Julian : TStDate ) : TStDayType
5912: Function DayOfWeekDMY( Day, Month, Year, Epoch : Integer ) : TStDayType
5913: Function StIsLeapYear( Year : Integer ) : Boolean
5914: Function StDaysInMonth( Month : Integer; Year, Epoch : Integer ) : Integer
5915: Function ResolveEpoch( Year, Epoch : Integer ) : Integer
5916: Function ValidTime( Hours, Minutes, Seconds : Integer ) : Boolean
5917: Procedure StTimeToHMS( T : TStTime; var Hours, Minutes, Seconds : Byte )
5918: Function HMStoStTime( Hours, Minutes, Seconds : Byte ) : TStTime
5919: Function CurrentTime : TStTime
5920: Procedure TimeDiff( Time1, Time2 : TStTime; var Hours, Minutes, Seconds : Byte )
5921: Function StIncTime( T : TStTime; Hours, Minutes, Seconds : Byte ) : TStTime
5922: Function DecTime( T : TStTime; Hours, Minutes, Seconds : Byte ) : TStTime
5923: Function RoundToNearestHour( T : TStTime; Truncate : Boolean ) : TStTime
5924: Function RoundToNearestMinute( const T : TStTime; Truncate : Boolean ) : TStTime
5925: Procedure DateTimeDiff( const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;var Days:LongInt;var Secs:LongInt )
5926: Procedure IncDateTime( const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;Days:Integer;Secs:LongInt )
5927: Function DateTimeToStDate( DT : TDateTime ) : TStDate
5928: Function DateTimeToStTime( DT : TDateTime ) : TStTime
5929: Function StDateToDate( D : TStDate ) : TDateTime
5930: Function StTimeToDate( T : TStTime ) : TDateTime
5931: Function Convert2ByteDate( TwoByteDate : Word ) : TStDate
5932: Function Convert4ByteDate( FourByteDate : TStDate ) : Word
5933:
5934: Procedure SIRegister_StDateSt(CL: TPSCompiler);
5935: Function DateStringHMStoAstJD( const Picture, DS : string; H, M, S, Epoch : integer ) : Double
5936: Function MonthToString( const Month : Integer ) : string
5937: Function DateStringToStDate( const Picture, S : string; Epoch : Integer ) : TStDate
5938: Function DateStringToDMY( const Picture,S:string;Epoch:Integer; var D, M, Y : Integer ):Boolean
5939: Function StDateToString( const Picture : string; const Julian : TStDate; Pack : Boolean):string
5940: Function DayOfWeekToString( const WeekDay : TStDayType ) : string
5941: Function DMYToDateString( const Picture:string;Day,Month,Year,Epoch:Integer;Pack:Boolean): string;
5942: Function CurrentDateString( const Picture : string; Pack : Boolean ) : string
5943: Function CurrentTimeString( const Picture : string; Pack : Boolean ) : string
5944: Function TimeStringToHMS( const Picture, St : string; var H, M, S : Integer ) : Boolean
5945: Function TimeStringToStTime( const Picture, S : string ) : TStTime
5946: Function StTimeToAmPmString( const Picture : string; const T : TStTime; Pack : Boolean ) : string
5947: Function StTimeToString( const Picture : string; const T : TStTime; Pack : Boolean ) : string
5948: Function DateStringIsBlank( const Picture, S : string ) : Boolean
5949: Function InternationalDate( ForceCentury : Boolean ) : string
5950: Function InternationalLongDate( ShortNames : Boolean; ExcludeDOW : Boolean ) : string
5951: Function InternationalTime( ShowSeconds : Boolean ) : string
5952: Procedure ResetInternationalInfo
5953:
5954: procedure SIRegister_StBase(CL: TPSCompiler);
5955: Function DestroyNode( Container : TStContainer; Node : TStNode; OtherData : Pointer ) : Boolean
5956: Function AnsiUpperCaseShort32( const S : string ) : string
5957: Function AnsiCompareTextShort32( const S1, S2 : string ) : Integer
5958: Function AnsiCompareStrShort32( const S1, S2 : string ) : Integer
5959: Function HugeCompressRLE( const InBuffer, InLen : Longint; var OutBuffer ) : Longint
5960: Function HugeDecompressRLE( const InBuffer, InLen : Longint; var OutBuffer,OutLen:LongInt): Longint
5961: Procedure HugeFillChar( var Dest, Count : Longint; Value : Byte )
5962: Procedure HugeFillStruc( var Dest, Count : Longint; const Value, ValSize : Cardinal )
5963: Function Upcase( C : AnsiChar ) : AnsiChar
5964: Function LoCase( C : AnsiChar ) : AnsiChar
5965: Function CompareLetterSets( Set1, Set2 : LongInt ) : Cardinal
5966: Function CompStruct( const S1, S2, Size : Cardinal ) : Integer
5967: Function Search( const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardinal):Bool;
5968: Function StSearch(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
5969: Function SearchUC(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
5970: Function IsOrInheritsFrom( Root, Candidate : TClass ) : boolean
5971: Procedure RaiseContainerError( Code : longint )
5972: Procedure RaiseContainerErrorFmt( Code : Longint; Data : array of const )
5973: Function ProductOverflow( A, B : LongInt ) : Boolean
5974: Function StNewStr( S : string ) : PShortString
5975: Procedure StDisposeStr( PS : PShortString )
5976: Procedure VaiLongInt( S : ShortString; var LI : Longint; var ErrorCode : integer )
5977: Procedure ValSmallint( const S : ShortString; var SI : smallint; var ErrorCode : integer )
5978: Procedure ValWord( const S : ShortString; var Wd : word; var ErrorCode : integer )
5979: Procedure RaiseStError( ExceptionClass : EStExceptionClass; Code : LongInt )
5980: Procedure RaiseStWin32Error( ExceptionClass : EStExceptionClass; Code : LongInt )
5981: Procedure RaiseStWin32ErrorEx( ExceptionClass : EStExceptionClass; Code : LongInt; Info : string )
5982:
5983: procedure SIRegister_usvd(CL: TPSCompiler);
5984: begin
5985: Procedure SV_DecomP( A : TMATRIX; Lb, Ub1, Ub2 : Integer; S : TVector; V : TMATRIX )
5986: Procedure SV_SetZero( S : TVector; Lb, Ub : Integer; Tol : Float )
5987: Procedure SV_Solve(U:TMATRIX; S:TVector;V:TMATRIX;B:TVector;Lb,Ub1,Ub2:Integer;X:TVector );

```

```

5988: Procedure SV_Approx( U : TMatrix; S : TVector; V : TMatrix; Lb, Ub1, Ub2 : Integer; A : TMatrix)
5989: Procedure RKF45(F:TDiffEqs;Neqn:Int;Y,Yp:TVector;var T:Float;Tout,RelErr,AbsErr:Float;var Flag:Int;
5990: end;
5991:
5992: //*****unit unit ; StMath Package of SysTools*****
5993: Function IntPowerS( Base : Extended; Exponent : Integer) : Extended
5994: Function PowerS( Base, Exponent : Extended) : Extended
5995: Function StInvCos( X : Double) : Double
5996: Function StInvSin( Y : Double) : Double
5997: Function StInvTan2( X, Y : Double) : Double
5998: Function StTan( A : Double) : Double
5999: Procedure DumpException; //unit STExpEng;
6000: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
6001:
6002: //*****unit unit ; StCRC Package of SysTools*****
6003: Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
6004: Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6005: Function Adler32OfFile( FileName : AnsiString) : LongInt
6006: Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6007: Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6008: Function Crc16OfFile( FileName : AnsiString) : Cardinal
6009: Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
6010: Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
6011: Function Crc32OfFile( FileName : AnsiString) : LongInt
6012: Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6013: Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
6014: Function InternetSumOfFile( FileName : AnsiString) : Cardinal
6015: Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
6016: Function Kermit16OfFile( Stream : TStream; CurCrc : Cardinal) : Cardinal
6017: Function Kermit16OfFile( FileName : AnsiString) : Cardinal
6018:
6019: //*****unit unit ; StBCD Package of SysTools*****
6020: Function AddBcd( const B1, B2 : Tbcds) : Tbcds
6021: Function SubBcd( const B1, B2 : Tbcds) : Tbcds
6022: Function MulBcd( const B1, B2 : Tbcds) : Tbcds
6023: Function DivBcd( const B1, B2 : Tbcds) : Tbcds
6024: Function ModBcd( const B1, B2 : Tbcds) : Tbcds
6025: Function NegBcd( const B : Tbcds) : Tbcds
6026: Function AbsBcd( const B : Tbcds) : Tbcds
6027: Function FracBcd( const B : Tbcds) : Tbcds
6028: Function IntBcd( const B : Tbcds) : Tbcds
6029: Function RoundDigitsBcd( const B : Tbcds; Digits : Cardinal) : Tbcds
6030: Function RoundPlacesBcd( const B : Tbcds; Places : Cardinal) : Tbcds
6031: Function ValBcd( const S : string) : Tbcds
6032: Function LongBcd( L : LongInt) : Tbcds
6033: Function ExtBcd( E : Extended) : Tbcds
6034: Function ExpBcd( const B : Tbcds) : Tbcds
6035: Function LnBcd( const B : Tbcds) : Tbcds
6036: Function IntPowBcd( const B : Tbcds; E : LongInt) : Tbcds
6037: Function PowBcd( const B, E : Tbcds) : Tbcds
6038: Function SqrtBcd( const B : Tbcds) : Tbcds
6039: Function CmpBcd( const B1, B2 : Tbcds) : Integer
6040: Function EqDigitsBcd( const B1, B2 : Tbcds; Digits : Cardinal) : Boolean
6041: Function EqPlacesBcd( const B1, B2 : Tbcds; Digits : Cardinal) : Boolean
6042: Function IsIntBcd( const B : Tbcds) : Boolean
6043: Function TruncBcd( const B : Tbcds) : LongInt
6044: Function BcdExt( const B : Tbcds) : Extended
6045: Function RoundBcd( const B : Tbcds) : LongInt
6046: Function StrBcd( const B : Tbcds; Width, Places : Cardinal) : string
6047: Function StrExpBcd( const B : Tbcds; Width : Cardinal) : string
6048: Function FormatBcd( const Format : string; const B : Tbcds) : string
6049: Function StrGeneralBcd( const B : Tbcds) : string
6050: Function FloatFormBcd(const Mask:string;B:Tbcds;const LtCurr,RtCurr:string;Sep,DecPt:AnsiChar):string
6051: Procedure ConvertBcd( const SrcB, SrcSize : Byte; var DestB, DestSize : Byte)
6052:
6053: //*****unit unit ; StTxtDat; TStTextDataRecordSet Package of SysTools*****
6054: Procedure StParseLine( const Data : AnsiString; Schema : TStTextDataSchema; Result : TStrings)
6055: Function StFieldTypeToStr( FieldType : TStSchemaFieldType) : AnsiString
6056: Function StStrToFieldType( const S : AnsiString) : TStSchemaFieldType
6057: Function StDeEscape( const EscStr : AnsiString) : Char
6058: Function StDoEscape( Delim : Char) : AnsiString
6059: Function StTrimTrailingChars( const S : AnsiString; Trailer : Char) : AnsiString
6060: Function AnsiHashText( const S : string; Size : Integer) : Integer
6061: Function AnsiHashStr( const S : string; Size : Integer) : Integer
6062: Function AnsiELFHashText( const S : string; Size : Integer) : Integer
6063: Function AnsiELFHashStr( const S : string; Size : Integer) : Integer
6064:
6065: //*****unit unit ; STNetCon Package of SysTools*****
6066: with AddClassN(FindClass('TStComponent'), 'TStNetConnection') do begin
6067:   Constructor Create( AOwner : TComponent)
6068:   Function Connect : DWord
6069:   Function Disconnect : DWord
6070:   RegisterProperty('Password', 'String', iptrw);
6071:   Property('UserName', 'String', iptrw);
6072:   Property('ConnectOptions', 'TStNetConnectOptionsSet', iptrw);
6073:   Property('DisconnectOptions', 'TStNetDisconnectOptionsSet', iptrw);
6074:   Property('LocalDevice', 'String', iptrw);
6075:   Property('ServerName', 'String', iptrw);
6076:   Property('ShareName', 'String', iptrw);

```

```

6077:   Property('OnConnect', 'TNotifyEvent', iptrw);
6078:   Property('OnConnectFail', 'TOnConnectFailEvent', iptrw);
6079:   Property('OnConnectCancel', 'TOnConnectCancelEvent', iptrw);
6080:   Property('OnDisconnect', 'TNotifyEvent', iptrw);
6081:   Property('OnDisconnectFail', 'TOnDisconnectFailEvent', iptrw);
6082:   Property('OnDisconnectCancel', 'TOnDisconnectCancelEvent', iptrw);
6083: end;
6084: //***** Thread Functions Context of Win API --- more objects in SyncObjs.pas
6085: / 153 unit uPSI_SyncObjs; unit uPSIParallelJobs;
6086: Procedure InitializeCriticalSection( var lpCriticalSection : TRTCriticalSection);
6087: Procedure EnterCriticalSection( var lpCriticalSection : TRTCriticalSection);
6088: Procedure LeaveCriticalSection( var lpCriticalSection : TRTCriticalSection);
6089: Function InitializeCriticalSectionAndSpinCount(var
6090: lpCriticalSection:TRTCriticalSection;dwSpinCount:DWORD):BOOL;
6091: Function SetCriticalSectionSpinCount(var lpCriticalSection:TRTCriticalSection;dwSpinCount:DWORD):DWORD;
6092: Function TryEnterCriticalSection( var lpCriticalSection : TRTCriticalSection) : BOOL;
6093: Procedure DeleteCriticalSection( var lpCriticalSection : TRTCriticalSection);
6094: Function GetThreadContext( hThread : THandle; var lpContext : TContext) : BOOL;
6095: Function SetThreadContext( hThread : THandle; const lpContext : TContext) : BOOL;
6096: Function SuspendThread( hThread : THandle) : DWORD;
6097: Function ResumeThread( hThread : THandle) : DWORD;
6098: Function CreateThread2(ThreadFunc: TThreadFunction2; thrid: DWord) : THandle;
6099: Function GetCurrentThread : THandle;
6100: Procedure ExitThread( dwExitCode : DWORD);
6101: Function TerminateThread( hThread : THandle; dwExitCode : DWORD) : BOOL;
6102: Procedure EndThread(ExitCode: Integer);
6103: Function WaitForSingleObject( hHandle : THandle; dwMilliseconds : DWORD) : DWORD;
6104: Function MakeProcInstance( Proc : FARPROC; Instance : THandle) : FARPROC;
6105: Procedure FreeProcInstance( Proc : FARPROC);
6106: Procedure FreeLibraryAndExitThread( hLibModule : HMODULE; dwExitCode : DWORD);
6107: Function DisableThreadLibraryCalls( hLibModule : HMODULE) : BOOL;
6108: Procedure ParallelJob( ASelf : TObject; ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6109: Procedure ParallelJob1(ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6110: Procedure ParallelJob2(AJobGroup:TJobsGroup;ASelf:TObject;ATarget:Ptr;AParam:Pointer;ASafeSection:bool);
6111: Procedure ParallelJob3( AJobGroup:TJobsGroup;ATarget:Pointer;AParam:Pointer;ASafeSection:boolean);
6112: Function CreateParallelJob(ASelf:TObject;ATarget:Pointer;AParam:Ptr;ASafeSection:bool):TParallelJob;
6113: Function CreateParallelJob1(ATarget:Pointer; AParam:Pointer; ASafeSection : boolean) : TParallelJob;
6114: Function CurrentParallelJobInfo : TParallelJobInfo;
6115: Function ObtainParallelJobInfo : TParallelJobInfo;
6116: Procedure GetSystemInfo( var lpSystemInfo : TSystemInfo)');
6117: Function IsProcessorFeaturePresent( ProcessorFeature : DWORD) : BOOL';
6118: Function SetStdHandle( nStdHandle : DWORD; hHandle : THandle) : BOOL';
6119: Function DeviceIoControl( hDevice : THandle; dwIoControlCode : DWORD; lpInBuffer : TObject; nInBufferSize : DWORD; lpOutBuffer: TObject; nOutBufferSize: DWORD; var lpBytesReturned: DWORD; lpOverlapped : TOverlapped) : BOOL';
6120: Function SetFileTime( hFile : THandle; lpCreationTime, lpLastAccessTime, lpLastWriteTime : TFileTime) : BOOL';
6121: Function DuplicateHandle( hSourceProcessHandle, hSourceHandle, hTargetProcessHandle : THandle; lpTargetHandle : THandle; dwDesiredAccess : DWORD; bInheritHandle : BOOL; dwOptions : DWORD) : BOOL';
6122: Function GetHandleInformation( hObject : THandle; var lpdwFlags : DWORD) : BOOL';
6123: Function SetHandleInformation( hObject : THandle; dwMask : DWORD; dwFlags : DWORD) : BOOL';
6124:
6125: *****unit uPSI_JclMime;
6126: Function MimeEncodeString( const S : AnsiString) : AnsiString;
6127: Function MimeDecodeString( const S : AnsiString) : AnsiString;
6128: Procedure MimeEncodeStream( const InputStream : TStream; const OutputStream : TStream);
6129: Procedure MimeDecodeStream( const InputStream : TStream; const OutputStream : TStream);
6130: Function MimeEncodedSize( const I : Cardinal) : Cardinal;
6131: Function MimeDecodedSize( const I : Cardinal) : Cardinal;
6132: Procedure MimeEncode( var InputBuffer : string; const InputByteCount : Cardinal; var OutputBuffer);
6133: Function MimeDecode(var InputBuffer:string;const InputBytesCount:Cardinal;var OutputBuffer):Cardinal;
6134: Function MimeDecodePartial(var InputBuffer:string;const InputBytesCount:Cardinal;var OutputBuffer:string;var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : Cardinal;
6135: Function MimeDecodePartialEnd(var OutputBuf:string;const ByteBuf:Card;const ByteBufferSpace:Card): Cardinal;
6136:
6137: *****unit uPSI_JclPrint;
6138: Procedure DirectPrint( const Printer, Data : string);
6139: Procedure SetPrinterPixelsPerInch;
6140: Function GetPrinterResolution : TPoint;
6141: Function CharFitsWithinDots( const Text : string; const Dots : Integer) : Integer;
6142: Procedure PrintMemo( const Memo : TMemo; const Rect : TRect);
6143:
6144:
6145: //*****unit uPSI_ShLwApi;
6146: Function StrChr( lpStart : PChar; wMatch : WORD) : PChar;
6147: Function StrChri( lpStart : PChar; wMatch : WORD) : PChar;
6148: Function StrCmpN( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer;
6149: Function StrCmpNI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer;
6150: Function StrCSpn( lpStr_, lpSet : PChar) : Integer;
6151: Function StrCSpnI( lpStr1, lpSet : PChar) : Integer;
6152: Function StrDup( lpSrch : PChar) : PChar;
6153: Function StrFormatByteSize( dw : DWORD; szBuf : PChar; uiBufSize : UINT) : PChar;
6154: Function StrFormatKBSIZE( qdw : Dword; szBuf : PChar; uiBufSize : UINT) : PChar;
6155: Function StrFromTimeInterval(pszOut: PChar; cchMax:UINT;dwTimeMS:DWORD; digits: Integer): Integer;
6156: Function StrIsIntLEqual( fCaseSens : Boolean; lpString1, lpString2 : PChar; nChar : Integer) : Boolean;
6157: /Function StrNCat( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar;
6158: Function StrPBrk( psz, pszSet : PChar) : PChar;

```

```

6159: Function StrRChr( lpStart, lpEnd : PChar; wMatch : WORD ) : PChar
6160: Function StrRChrI( lpStart, lpEnd : PChar; wMatch : WORD ) : PChar
6161: Function StrRStrI( lpSource, lpLast, lpSrch : PChar ) : PChar
6162: Function StrSpn( psz, pszSet : PChar ) : Integer
6163: Function StrStr( lpFirst, lpSrch : PChar ) : PChar
6164: Function StrStrI( lpFirst, lpSrch : PChar ) : PChar
6165: Function StrToInt( lpSrch : PChar ) : Integer
6166: Function StrToIntEx( pszString : PChar; dwFlags : DWORD; var piRet : Integer ) : BOOL
6167: Function StrTrim( psz : PChar; pszTrimChars : PChar ) : BOOL
6168: Function ChrCmpI( w1, w2 : WORD ) : BOOL
6169: Function ChrCmpIA( w1, w2 : WORD ) : BOOL
6170: Function ChrCmpIW( w1, w2 : WORD ) : BOOL
6171: Function StrIntEqN( s1, s2 : PChar; nChar : Integer ) : BOOL
6172: Function StrIntEqNI( s1, s2 : PChar; nChar : Integer ) : BOOL
6173: Function StrCatBuff( pszDest, pszSrc : PChar; cchDestBuffSize : Integer ) : PChar
6174: Function StrCpyNX( psz1 : PChar; psz2 : PChar; cchMax : Integer ) : PChar
6175: Function IntlStrEqWorker( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer ) : BOOL
6176: Function IntlStrEqN( s1, s2 : PChar; nChar : Integer ) : BOOL
6177: SZ_CONTENTTYPE_HTMLA', 'String 'text/html'
6178: SZ_CONTENTTYPE_HTMLW', 'String 'text/html'
6179: SZ_CONTENTTYPE_HTML', 'string SZ_CONTENTTYPE_HTMLA';
6180: SZ_CONTENTTYPE_CDFA', 'String 'application/x-cdf'
6181: SZ_CONTENTTYPE_CDFW', 'String 'application/x-cdf'
6182: SZ_CONTENTTYPE_CDF', 'string SZ_CONTENTTYPE_CDFA';
6183: Function PathIsHTMLFile( pszPath : PChar ) : BOOL
6184: STIF_DEFAULT', 'LongWord( $00000000);
6185: STIF_SUPPORT_HEX', 'LongWord( $00000001);
6186: Function StrNCmpI( lpStr1, lpStr2 : PChar; nChar : Integer ) : Integer
6187: Function StrNCopy( psz1, psz2 : PChar; cchMax : Integer ) : PChar
6188: Function StrCatN( psz1 : PChar; psz2 : PChar; cchMax : Integer ) : PChar
6189: Function PathAddBackslash( pszPath : PChar ) : PChar
6190: Function PathAddExtension( pszPath : PChar; pszExt : PChar ) : BOOL
6191: Function PathAppend( pszPath : PChar; pMore : PChar ) : BOOL
6192: Function PathBuildRoot( szRoot : PChar; iDrive : Integer ) : PChar
6193: Function PathCanonicalize( pszBuf : PChar; pszPath : PChar ) : BOOL
6194: Function PathCombine( szDest : PChar; lpszDir, lpszFile : PChar ) : PChar
6195: Function PathCompactPath( hDC : HDC; pszPath : PChar; dx : UINT ) : BOOL
6196: Function PathCompactPathEx( pszOut : PChar; pszSrc : PChar; cchMax : UINT; dwFlags:DWORD ) : BOOL
6197: Function PathCommonPrefix( pszFile1, pszFile2 : PChar; achPath : PChar ) : Integer
6198: Function PathFileExists( pszPath : PChar ) : BOOL
6199: Function PathFindExtension( pszPath : PChar ) : PChar
6200: Function PathFindFileName( pszPath : PChar ) : PChar
6201: Function PathFindNextComponent( pszPath : PChar ) : PChar
6202: Function PathFindOnPath( pszPath : PChar; var ppszOtherDirs : PChar ) : BOOL
6203: Function PathGetArgs( pszPath : PChar ) : PChar
6204: Function PathFindSuffixArray(pszPath: PChar; const apszSuffix: PChar; iArraySize: Integer): PChar
6205: Function PathIsLFNFileSpec( lpName : PChar ) : BOOL
6206: Function PathGetCharType( ch : Char ) : UINT
6207: GCT_INVALID', 'LongWord( $0000);
6208: GCT_LFNCHAR', 'LongWord( $0001);
6209: GCT_SHORTCHAR', 'LongWord( $0002);
6210: GCT_WILD', 'LongWord( $0004);
6211: GCT_SEPARATOR', 'LongWord( $0008);
6212: Function PathGetDriveNumber( pszPath : PChar ) : Integer
6213: Function PathIsDirectory( pszPath : PChar ) : BOOL
6214: Function PathIsDirectoryEmpty( pszPath : PChar ) : BOOL
6215: Function PathIsFileSpec( pszPath : PChar ) : BOOL
6216: Function PathIsPrefix( pszPrefix, pszPath : PChar ) : BOOL
6217: Function PathIsRelative( pszPath : PChar ) : BOOL
6218: Function PathIsRoot( pszPath : PChar ) : BOOL
6219: Function PathIsSameRoot( pszPath1, pszPath2 : PChar ) : BOOL
6220: Function PathIsUNC( pszPath : PChar ) : BOOL
6221: Function PathIsNetworkPath( pszPath : PChar ) : BOOL
6222: Function PathIsUNCServer( pszPath : PChar ) : BOOL
6223: Function PathIsUNCServerShare( pszPath : PChar ) : BOOL
6224: Function PathIsContentType( pszPath, pszContentType : PChar ) : BOOL
6225: Function PathIsURL( pszPath : PChar ) : BOOL
6226: Function PathMakePretty( pszPath : PChar ) : BOOL
6227: Function PathMatchSpec( pszFile, pszSpec : PChar ) : BOOL
6228: Function PathParseIconLocation( pszIconFile : PChar ) : Integer
6229: Procedure PathQuoteSpaces( lpsz : PChar )
6230: Function PathRelativePathTo(pszPath:PChar;pszFrom:PChar;dwAttrFrom:DWORD;pszTo:PChar;dwAttrTo:DWORD):BOOL;
6231: Procedure PathRemoveArgs( pszPath : PChar )
6232: Function PathRemoveBackslash( pszPath : PChar ) : PChar
6233: Procedure PathRemoveBlanks( pszPath : PChar )
6234: Procedure PathRemoveExtension( pszPath : PChar )
6235: Function PathRemoveFileSpec( pszPath : PChar ) : BOOL
6236: Function PathRenameExtension( pszPath : PChar; pszExt : PChar ) : BOOL
6237: Function PathSearchAndQualify( pszPath : PChar; pszBuf : PChar; cchBuf : UINT ) : BOOL
6238: Procedure PathSetDlgItemPath( hDlg : HWND; id : Integer; pszPath : PChar )
6239: Function PathSkipRoot( pszPath : PChar ) : PChar
6240: Procedure PathStripPath( pszPath : PChar )
6241: Function PathStripToRoot( pszPath : PChar ) : BOOL
6242: Procedure PathUnquoteSpaces( lpsz : PChar )
6243: Function PathMakeSystemFolder( pszPath : PChar ) : BOOL
6244: Function PathUnmakeSystemFolder( pszPath : PChar ) : BOOL
6245: Function PathIsSystemFolder( pszPath : PChar; dwAttrib : DWORD ) : BOOL
6246: Procedure PathUndecorate( pszPath : PChar )
6247: Function PathUnExpandEnvStrings( pszPath : PChar; pszBuf : PChar; cchBuf : UINT ) : BOOL

```

```

6248: URL_SCHEME_INVALID', 'LongInt'( - 1);
6249: URL_SCHEME_UNKNOWN', 'LongInt'( 0);
6250: URL_SCHEME_FTP', 'LongInt'( 1);
6251: URL_SCHEME_HTTP', 'LongInt'( 2);
6252: URL_SCHEME_GOPHER', 'LongInt'( 3);
6253: URL_SCHEME_MAILTO', 'LongInt'( 4);
6254: URL_SCHEME_NEWS', 'LongInt'( 5);
6255: URL_SCHEME_NNTP', 'LongInt'( 6);
6256: URL_SCHEME_TELNET', 'LongInt'( 7);
6257: URL_SCHEME_WAIS', 'LongInt'( 8);
6258: URL_SCHEME_FILE', 'LongInt'( 9);
6259: URL_SCHEME_MK', 'LongInt'( 10);
6260: URL_SCHEME_HTTPS', 'LongInt'( 11);
6261: URL_SCHEME_SHLL', 'LongInt'( 12);
6262: URL_SCHEME_SNEWS', 'LongInt'( 13);
6263: URL_SCHEME_LOCAL', 'LongInt'( 14);
6264: URL_SCHEME_JAVASCRIPT', 'LongInt'( 15);
6265: URL_SCHEME_VBSCRIPT', 'LongInt'( 16);
6266: URL_SCHEME_ABOUT', 'LongInt'( 17);
6267: URL_SCHEME_RES', 'LongInt'( 18);
6268: URL_SCHEME_MAXVALUE', 'LongInt'( 19);
6269: URL_SCHEME', 'Integer
6270: URL_PART_NONE', 'LongInt'( 0);
6271: URL_PART_SCHEME', 'LongInt'( 1);
6272: URL_PART_HOSTNAME', 'LongInt'( 2);
6273: URL_PART_USERNAME', 'LongInt'( 3);
6274: URL_PART_PASSWORD', 'LongInt'( 4);
6275: URL_PART_PORT', 'LongInt'( 5);
6276: URL_PART_QUERY', 'LongInt'( 6);
6277: URL_PART', 'DWORD
6278: URLIS_URL', 'LongInt'( 0);
6279: URLIS_OPAQUE', 'LongInt'( 1);
6280: URLIS_NOHISTORY', 'LongInt'( 2);
6281: URLIS_FILEURL', 'LongInt'( 3);
6282: URLIS_APPLICABLE', 'LongInt'( 4);
6283: URLIS_DIRECTORY', 'LongInt'( 5);
6284: URLIS_HASQUERY', 'LongInt'( 6);
6285: TURLIs', 'DWORD
6286: URL_UNESCAPE', 'LongWord( $10000000);
6287: URL_ESCAPE_UNSAFE', 'LongWord( $20000000);
6288: URL_PLUGGABLE_PROTOCOL', 'LongWord( $40000000);
6289: URL_WININET_COMPATIBILITY', 'LongWord( DWORD( $80000000 ) );
6290: URL_DONT_ESCAPE_EXTRA_INFO', 'LongWord( $02000000);
6291: URL_ESCAPE_SPACES_ONLY', 'LongWord( $04000000);
6292: URL_DONT_SIMPLIFY', 'LongWord( $08000000);
6293: URL_NO_META', 'longword( URL_DONT_SIMPLIFY);
6294: URL_UNESCAPE_INPLACE', 'LongWord( $00100000);
6295: URL_CONVERT_IF_DOSPATH', 'LongWord( $00200000);
6296: URL_UNESCAPE_HIGH_ANSI_ONLY', 'LongWord( $00400000);
6297: URL_INTERNAL_PATH', 'LongWord( $00800000);
6298: URL_FILE_USE_PATHURL', 'LongWord( $00010000);
6299: URL_ESCAPE_PERCENT', 'LongWord( $00001000);
6300: URL_ESCAPE_SEGMENT_ONLY', 'LongWord( $00002000);
6301: URL_PARTFLAG_KEEPSCHEME', 'LongWord( $00000001);
6302: URL_APPLY_DEFAULT', 'LongWord( $00000001);
6303: URL_APPLY_GUESSSCHEME', 'LongWord( $00000002);
6304: URL_APPLY_GUESSFILE', 'LongWord( $00000004);
6305: URL_APPLY_FORCEAPPLY', 'LongWord( $00000008);
6306: Function UrlCompare( psz1, psz2 : PChar; fIgnoreSlash : BOOL) : Integer
6307: Function UrlCombine(pszBase,pszRelative:PChar;pszCombin:PChar;out pcchCombin:DWORD;dwFlags:DWORD) : HRESULT;
6308: Function UrlCanonicalize(pszUrl:PChar;pszCanonicalized:PChar;pcchCanonic:DWORD;dwFlags:DWORD) : HRESULT;
6309: Function UrlIsOpaque( pszURL : PChar) : BOOL
6310: Function UrlIsNoHistory( pszURL : PChar) : BOOL
6311: Function UrlIsFileUrl( pszURL : PChar) : BOOL
6312: Function UrlIs( pszUrl : PChar; UrlIs : TURLIs) : BOOL
6313: Function UrlGetLocation( psz1 : PChar) : PChar
6314: Function UrlUnescape( pszUrl, pszUnescaped : PChar;pcchUnescaped:DWORD; dwFlags : DWORD) : HRESULT
6315: Function UrlEscape(pszUrl: PChar; pszEscaped: PChar; pcchEscaped:DWORD; dwFlags : DWORD) : HRESULT
6316: Function UrlCreateFromPath(pszPath:PChar; pszUrl: PChar;pcchUrl: DWORD; dwFlags : DWORD) : HRESULT
6317: Function PathCreateFromUrl(pszUrl:PChar; pszPath:PChar; pcchPath:DWORD; dwFlags : DWORD) : HRESULT
6318: Function UrlHash( pszUrl : PChar; pbHash : BYTE; cbHash : DWORD) : HRESULT
6319: Function UrlGetPart(pszIn: PChar; pszOut: PChar; pcchOut: DWORD; dwPart,dwFlags: DWORD) : HRESULT
6320: Function UrlApplyScheme( pszIn : PChar; pszOut : PChar; pcchOut : DWORD; dwFlags : DWORD) : HRESULT
6321: Function HashData( pData : BYTE; cbData : DWORD; pbHash : BYTE; cbHash : DWORD) : HRESULT
6322: Function UrlEscapeSpaces( pszUrl : PChar; pszEscaped : PChar; pcchEscaped : DWORD) : HRESULT
6323: Function UrlUnescapeInPlace( pszUrl : PChar; dwFlags : DWORD) : HRESULT
6324: Function SHDeleteEmptyKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6325: Function SHDeleteKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6326: Function SHDeleteValue( hKey : HKEY; pszSubKey, pszValue : PChar) : DWORD
6327: Function SHEnumKeyEx( hKey : HKEY; dwIndex : DWORD; pszName : PChar; var pcchName : DWORD) : Longint
6328: Function SHEnumValue( hKey : HKEY; dwIndex : DWORD; pszValueName:PChar; var
pcchValueName:DWORD;pdwType:DWORD; pvData : Pointer; pcbData : DWORD) : Longint
6329: Function SHQueryInfoKey(hKey:HKEY;pcSubKeys,pcchMaxSubKeyLen,pcVal,pcch.MaxValueNameLen:DWORD):Longint;
6330: Function SHCopyKey( hkeySrc : HKEY; szSrcSubKey : PChar; hkeyDest : HKEY; fReserved : DWORD) : DWORD
6331: Function SHRegGetPath(hKey:HKEY; ppszSubKey,ppszValue: PChar; pszPath: PChar; dwFlags: DWORD): DWORD
6332: Function SHRegSetPath( hKey:HKEY; ppszSubKey, ppszValue, ppszPath : PChar; dwFlags : DWORD): DWORD
6333: SHREGDEL_DEFAULT', 'LongWord( $00000000);
6334: SHREGDEL_HKCU', 'LongWord( $00000001);
6335: SHREGDEL_HKLM', 'LongWord( $00000010);

```

```

6336: SHREGDEL_BOTH', 'LongWord( $00000011);
6337: SHREGENUM_DEFAULT', 'LongWord( $00000000);
6338: SHREGENUM_HKCU', 'LongWord( $00000001);
6339: SHREGENUM_HKLM', 'LongWord( $00000010);
6340: SHREGENUM_BOTH', 'LongWord( $00000011);
6341: SHREGSET_HKCU', 'LongWord( $00000001);
6342: SHREGSET_FORCE_HKCU', 'LongWord( $00000002);
6343: SHREGSET_HKLM', 'LongWord( $00000004);
6344: SHREGSET_FORCE_HKLM', 'LongWord( $00000008);
6345: TSHRegDelFlags', 'DWORD
6346: TSHRegEnumFlags', 'DWORD
6347: HUSKEY', 'THandle
6348: ASSOCF_INIT_NOREMAPCLSID', 'LongWord( $00000001);
6349: ASSOCF_INIT_BYEXENAME', 'LongWord( $00000002);
6350: ASSOCF_OPEN_BYEXENAME', 'LongWord( $00000002);
6351: ASSOCF_INIT_DEFAULTTOSTAR', 'LongWord( $00000004);
6352: ASSOCF_INIT_DEFAULTTOFOLDER', 'LongWord( $00000008);
6353: ASSOCF_NOUSERSETTINGS', 'LongWord( $00000010);
6354: ASSOCF_NOTRUNCATE', 'LongWord( $00000020);
6355: ASSOCF_VERIFY', 'LongWord( $00000040);
6356: ASSOCF_REMAPRUNDLL', 'LongWord( $00000080);
6357: ASSOCF_NOFIXUPS', 'LongWord( $00000100);
6358: ASSOCF_IGNOREBASECLASS', 'LongWord( $00000200);
6359: ASSOCF', 'DWORD
6360: ASSOCSTR_COMMAND', 'LongInt'( 1);
6361: ASSOCSTR_EXECUTABLE', 'LongInt'( 2);
6362: ASSOCSTR_FRIENDLYDOCNAME', 'LongInt'( 3);
6363: ASSOCSTR_FRIENDLYAPPNAME', 'LongInt'( 4);
6364: ASSOCSTR_NOOPEN', 'LongInt'( 5);
6365: ASSOCSTR_SHELLNEWVALUE', 'LongInt'( 6);
6366: ASSOCSTR_DDECOMMAND', 'LongInt'( 7);
6367: ASSOCSTR_DDEIFEXEC', 'LongInt'( 8);
6368: ASSOCSTR_DDEAPPLICATION', 'LongInt'( 9);
6369: ASSOCSTR_DDETOPIC', 'LongInt'( 10);
6370: ASSOCSTR_INFOTIP', 'LongInt'( 11);
6371: ASSOCSTR_MAX', 'LongInt'( 12);
6372: ASSOCSTR', 'DWORD
6373: ASSOCKEY_SHELLEXECCLASS', 'LongInt'( 1);
6374: ASSOCKEY_APP', 'LongInt'( 2);
6375: ASSOCKEY_CLASS', 'LongInt'( 3);
6376: ASSOCKEY_BASECLASS', 'LongInt'( 4);
6377: ASSOCKEY_MAX', 'LongInt'( 5);
6378: ASSOCKEY', 'DWORD
6379: ASSOCDATA_MSIDEScriptor', 'LongInt'( 1);
6380: ASSOCDATA_NOACTIVATEHANDLER', 'LongInt'( 2);
6381: ASSOCDATA_QUERYCLASSTORE', 'LongInt'( 3);
6382: ASSOCDATA_HASPERUSERASSOC', 'LongInt'( 4);
6383: ASSOCDATA_MAX', 'LongInt'( 5);
6384: ASSOCDATA', 'DWORD
6385: ASSOCENUM_NONE', 'LongInt'( 0);
6386: ASSOCENUM', 'DWORD
6387: SID_IQueryAssociations', 'String' '{c46ca590-3c3f-11d2-bee6-0000f805ca57}
6388: SHACF_DEFAULT $00000000;
6389: SHACF_FILESYSTEM', 'LongWord( $00000001);
6390: SHACF_URLHISTORY', 'LongWord( $00000002);
6391: SHACF_URLMRU', 'LongWord( $00000004);
6392: SHACF_USETAB', 'LongWord( $00000008);
6393: SHACF_FILESYS_ONLY', 'LongWord( $00000010);
6394: SHACF_AUTOSUGGEST_FORCE_ON', 'LongWord( $10000000);
6395: SHACF_AUTOSUGGEST_FORCE_OFF', 'LongWord( $20000000);
6396: SHACF_AUTOAPPEND_FORCE_ON', 'LongWord( $40000000);
6397: SHACF_AUTOAPPEND_FORCE_OFF', 'LongWord( DWORD ( $80000000 ) );
6398: Function SHAutoComplete( hwndEdit : HWND; dwFlags : DWORD ) : HRESULT
6399: Procedure SHSetThreadRef( punk : IUnknown )
6400: Procedure SHGetThreadRef( out ppunk : IUnknown )
6401: CTF_INSIST', 'LongWord( $00000001);
6402: CTF_THREAD_REF', 'LongWord( $00000002);
6403: CTF_PROCESS_REF', 'LongWord( $00000004);
6404: CTF_COINIT', 'LongWord( $00000008);
6405: Function SHCreateShellPalette( hdc : HDC ) : HPALETTE
6406: Procedure ColorRGBToHLS( clrRGB : TColorRef; out pwHue, pwLuminance, pwSaturation : WORD )
6407: Function ColorHLSTORGB( wHue, wLuminance, wSaturation : WORD ) : TColorRef
6408: Function ColorAdjustLuma( clrRGB : TColorRef; n : Integer; fScale : Boolean ) : TColorRef
6409: Function GetSysColorBrush( nIndex : Integer ) : HBRUSH
6410: Function DrawFocusRect( hdc : HDC; const lprc : TRect ) : BOOL
6411: Function FillRect( hdc : HDC; const lprc : TRect; hbr : HBRUSH ) : Integer
6412: Function FrameRect( hdc : HDC; const lprc : TRect; hbr : HBRUSH ) : Integer
6413: Function InvertRect( hdc : HDC; const lprc : TRect ) : BOOL
6414: Function SetRect( var lprc : TRect; xLeft, yTop, xRight, yBottom : Integer ) : BOOL
6415: Function SetRectEmpty( var lprc : TRect ) : BOOL
6416: Function CopyRect( var lprcDst : TRect; const lprcSrc : TRect ) : BOOL
6417: Function InflateRect( var lprc : TRect; dx, dy : Integer ) : BOOL
6418: Function IntersectRect2( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect ) : BOOL
6419: Function SubtractRect( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect ) : BOOL
6420:
6421: Function InitializeFlatSB( hWnd : HWND ) : Bool
6422: Procedure UninitializeFlatSB( hWnd : HWND )
6423: Function FlatSB_GetScrollProp( pl : HWND; propIndex : Integer; p3 : PInteger ) : Bool
6424: Function FlatSB_SetScrollProp( pl : HWND; index : Integer; newValue : Integer; p4 : Bool ) : Bool

```

```

6425: Function GET_APPCOMMAND_LPARAM( lParam : Integer ) : Word //of JvWin32
6426: Function GET_DEVICE_LPARAM( lParam : Integer ) : Word
6427: Function GET_MOUSEORKEY_LPARAM( lParam : Integer ) : Integer
6428: Function GET_FLAGS_LPARAM( lParam : Integer ) : Word
6429: Function GET_KEYSTATE_LPARAM( lParam : Integer ) : Word
6430:
6431:
6432: // **** 204 unit uPSI_ShellAPI;
6433: Function DragQueryFile( Drop : HDROP; FileIndex : UINT; FileName : PChar; cb : UINT ) : UINT
6434: Function DragQueryPoint( Drop : HDROP; var Point : TPoint ) : BOOL
6435: Procedure DragFinish( Drop : HDROP )
6436: Procedure DragAcceptFiles( Wnd : HWND; Accept : BOOL )
6437: Function ShellExecute( hWnd : HWND; Operation,FileName,Parameters,Directory : PChar; ShowCmd : Integer ) : HINST
6438: Function FindExecutable( FileName, Directory : PChar; Result : PChar ) : HINST
6439: Function ShellAbout( Wnd : HWND; szApp, szOtherStuff : PChar; Icon : HICON ) : Integer
6440: Function DuplicateIcon( hInst : HINST; Icon : HICON ) : HICON
6441: Function ExtractAssociatedIcon( hInst : HINST; lpIconPath : PChar; var lpiIcon : Word ) : HICON
6442: Function ExtractIcon( hInst : HINST; lpszExeFileName : PChar; nIconIndex : UINT ) : HICON
6443: Function SHAppBarMessage( dwMessage : DWORD; var pData : TAppBarData ) : UINT
6444: Function DoEnvironmentSubst( szString : PChar; cbString : UINT ) : DWORD
6445: Function ExtractIconEx( lpszFile : PChar; nIconIndex : Int; var phiconLarge,phiconSmall : HICON; nIcons : UINT ) : UINT
6446: Procedure SHFreeNameMappings( hNameMappings : THandle )
6447:
6448: DLLVER_PLATFORM_WINDOWS', 'LongWord( $00000001 );
6449: DLLVER_PLATFORM_NT', 'LongWord( $00000002 );
6450: DLLVER_MAJOR_MASK', 'LongWord( Int64( $FFFF000000000000 ) );
6451: DLLVER_MINOR_MASK', 'LongWord( Int64( $0000FFFF00000000 ) );
6452: DLLVER_BUILD_MASK', 'LongWord( Int64( $00000000FFFF0000 ) );
6453: DLLVER_QFE_MASK', 'LongWord( Int64( $000000000000FFFF ) );
6454: Function MAKEDLLVERULL( Major, Minor, Build, Qfe : Word ) : Int64
6455: Function SimpleXMLEncode( const S : string ) : string
6456: Procedure SimpleXMLDecode( var S : string; TrimBlanks : Boolean )
6457: Function XMLEncode( const S : string ) : string
6458: Function XMLDecode( const S : string ) : string
6459: Function EntityEncode( const S : string ) : string
6460: Function EntityDecode( const S : string ) : string
6461:
6462: procedure RIRegister_CPort_Routines( S : TPSEExec );
6463: Procedure EnumComPorts( Ports : TStrings )
6464: Procedure ListComPorts( Ports : TStrings )
6465: Procedure ComPorts( Ports : TStrings ) //Alias to Arduino
6466: Function GetComPorts : TStringlist;
6467: Function StrToBaudRate( Str : string ) : TBaudRate
6468: Function StrToStopBits( Str : string ) : TStopBits
6469: Function StrToDataBits( Str : string ) : TDataBits
6470: Function StrToParity( Str : string ) : TParityBits
6471: Function StrToFlowControl( Str : string ) : TFlowControl
6472: Function BaudRateToStr( BaudRate : TBaudRate ) : string
6473: Function StopBitsToStr( StopBits : TStopBits ) : string
6474: Function DataBitsToStr( DataBits : TDataBits ) : string
6475: Function ParityToStr( Parity : TParityBits ) : string
6476: Function FlowControlToStr( FlowControl : TFlowControl ) : string
6477: Function ComErrorsToStr( Errors : TComErrors ) : String
6478:
6479: Function GetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT ) : BOOL
6480: Function DispatchMessage( const lpMsg : TMsg ) : Longint
6481: Function TranslateMessage( const lpMsg : TMsg ) : BOOL
6482: Function SetMessageQueue( cMessagesMax : Integer ) : BOOL
6483: Function PeekMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax, wRemoveMsg : UINT ) : BOOL
6484: Function GetMessagePos : DWORD
6485: Function GetMessageTime : Longint
6486: Function GetMessageExtraInfo : Longint
6487: Function GetSpecialFolderPath( const FolderName : string; CanCreate : Boolean ) : string
6488: Procedure JAddToRecentDocs( const Filename : string )
6489: Procedure ClearRecentDocs
6490: Function ExtractIconFromFile( FileName : string; Index : Integer ) : HICON
6491: Function CreateShellLink( const AppName, Desc : string; Dest : string ) : string
6492: Procedure GetShellLinkInfo( const LinkFile : WideString; var SLI : TShellLinkInfo )
6493: Procedure SetShellLinkInfo( const LinkFile : WideString; const SLI : TShellLinkInfo )
6494: Function RecycleFile( FileToRecycle : string ) : Boolean
6495: Function JCopyFile( FromFile, ToDir : string ) : Boolean
6496: Function ShellObjectTypeEnumToConst( ShellObjectType : TShellObjectType ) : UInt
6497: Function ShellObjectTypeConstToEnum( ShellObjectType : UInt ) : TShellObjectType
6498: Function QueryServiceConfig2A( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded : DWORD ) : Boolean
6499: Function QueryServiceConfig2W( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded : DWORD ) : Boolean
6500: Function QueryServiceConfig2( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded : DWORD ) : Boolean
6501: Function EnumServicesStatusExA( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD; dwServiceState : DWORD; lpServices : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned, lpResumeHandle : DWORD; pszGroupName : LP
6502: Function EnumServicesStatusExW( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD; dwServiceState : DWORD; lpServices : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned, lpResumeHandle : DWORD; pszGroupName : LP
6503: Function EnumServicesStatusEx( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD; dwServiceState : DWORD; lpServices : LPBYTE; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned, lpResumeHandle : DWORD; pszGroupName : LP
6504: Function ConvertSidToStringSid( sid : PSID; var stringSid : LPWSTR ) : Boolean

```

```

6505:
6506: ***** unit uPSI_JclPeImage;
6507:
6508: Function IsValidPeFile( const FileName : TFileName ) : Boolean
6509: // Function PeGetNtHeaders( const FileName : TFileName; var NtHeaders : TImageNtHeaders ) : Boolean
6510: Function PeCreateNameHintTable( const FileName : TFileName ) : Boolean
6511: Function PeRebaseImage(const ImageName: TFileName; NewBase : DWORD; TimeStamp : DWORD; MaxNewSize : DWORD) : TJclRebaseImageInfo
6512: Function PeVerifyCheckSum( const FileName : TFileName ) : Boolean
6513: Function PeClearCheckSum( const FileName : TFileName ) : Boolean
6514: Function PeUpdateCheckSum( const FileName : TFileName ) : Boolean
6515: Function PeDoesExportFunction(const FileName:TFileName;const FuncName:string;Options:TJclSmartCompOptions):Bool;
6516: Function PeIsExportFunctionForwardedEx( const FileName : TFileName; const FunctionName : string; var ForwardedName : string; Options : TJclSmartCompOptions ) : Boolean
6517: Function PeIsExportFunctionForwarded(const FileName:TFileName;const FunctionName:string;Options:TJclSmartCompOptions):Bool
6518: Function PeDoesImportFunction( const FileName : TFileName; const FunctionName : string; const LibraryName : string; Options : TJclSmartCompOptions ) : Boolean
6519: Function PeDoesImportLibrary(const FileName:TFileName;const LibraryName:string;Recursive:Boolean):Boolean;
6520: Function PeImportedLibraries( const FileName : TFileName; const LibrariesList : TStrings; Recursive : Boolean; FullPathName : Boolean ) : Boolean
6521: Function PeImportedFunctions(const FileName:TFileName;const FunctionsList:TStrings;const LibraryName:string;IncludeLibNames : Boolean): Boolean
6522: Function PeExportedFunctions( const FileName : TFileName; const FunctionsList : TStrings ) : Boolean
6523: Function PeExportedNames( const FileName : TFileName; const FunctionsList : TStrings ) : Boolean
6524: Function PeExportedVariables( const FileName : TFileName; const FunctionsList : TStrings ) : Boolean
6525: Function PeResourceKindNames(const FileN: TFileName;ResourceType:TJclPeResourceKind;const NamesList:TStrings):Bool
6526: Function PeBorFormNames( const FileName : TFileName; const NamesList : TStrings ) : Boolean
6527: Function PeBorDependedPackages(const FileName:TFileName;PackagesList:TStrings;FullPathName, Descript:Bool):Bool;
6528: Function PeFindMissingImports( const FileName : TFileName; MissingImportsList : TStrings ) : Boolean;
6529: Function PeFindMissingImports1( RequiredImportsList, MissingImportsList : TStrings ) : Boolean;
6530: Function PeCreateRequiredImportList(const FileName : TFileName; RequiredImportsList: TStrings): Boolean;
6531: //Function PeMapImgNtHeaders( const BaseAddress : Pointer ) : PImageNtHeaders
6532: //Function PeMapImgLibraryName( const BaseAddress : Pointer ) : string
6533: //Function PeMapImgSections( const NtHeaders : PImageNtHeaders ) : PImageSectionHeader
6534: //Function PeMapImgFindSection( const NtHeaders : PImageNtHeaders; const SectionName : string ) : PImageSectionHeader
6535: //Function PeMapImgExportedVariables(const Module: HMODULE; const VariablesList:TStrings):Boolean
6536: //Function PeMapImgResolvePackageThunk( Address : Pointer ) : Pointer
6537: Function PeMapFindResource(const Module:HMODULE;const ResourceType:PChar;const ResourceName:string):__Pointer;
6538: SIRegister_TJclPeSectionStream(CL);
6539: SIRegister_TJclPeMapImgHookItem(CL);
6540: SIRegister_TJclPeMapImgHooks(CL);
6541: //Function PeDbgImgNtHeaders(ProcessHandle:THandle;BaseAddress:Pointer,var NtHeaders:TImageNtHeaders):Boolean
6542: //Function PeDdbgImgLibraryName(ProcessHandle:THandle; BaseAddress:Pointer; var Name:string):Boolean
6543: Type TJclBorUmSymbolKind', '(skData,skFunction,skConstructor,skDestructor,skRTTI,skVTable)
6544: TJclBorUmSymbolModifier', '( smQualified, smLinkProc )
6545: TJclBorUmSymbolModifiers', 'set of TJclBorUmSymbolModifier
6546: TJclBorUmDescription', 'record Kind : TJclBorUmSymbolKind; Modifiers : TJclBorUmSymbolModifiers; end
6547: TJclBorUmResult', '( urOk, urNotMangled, urMicrosoft, urError )
6548: TJclPeUmResult', '( umNotMangled, umBorland, umMicrosoft )
6549: Function PeBorUmUnmangleName( const Name : string; var Unmangled : string; var Description : TJclBorUmDescription; var BasePos : Integer ) : TJclBorUmResult;
6550: Function PeBorUmUnmangleName1(const Name:string;var Unmangled:string;var Descript:TJclBorUmDescription):TJclBorUmResult;
6551: Function PeBorUmUnmangleName2( const Name : string; var Unmangled : string ) : TJclBorUmResult;
6552: Function PeBorUmUnmangleName3( const Name : string ) : string;
6553: Function PeIsNameMangled( const Name : string ) : TJclPeUmResult
6554: Function PeUnmangleName( const Name : string; var Unmangled : string ) : TJclPeUmResult
6555:
6556:
6557: //***** SysTools uPSI_StSystem; *****
6558: Function StCopyFile( const SrcPath, DestPath : AnsiString ) : Cardinal
6559: Function CreateTempFile( const aFolder : AnsiString; const aPrefix : AnsiString ) : AnsiString
6560: Function DeleteVolumeLabel( Drive : Char ) : Cardinal
6561: //Procedure EnumerateDirectories(const StartDir:AnsiStr;FL:TStrings;SubDirs:Bool;IncludeItem:TIncludeItemFunc);
6562: //Procedure EnumerateFiles(const StartDir:AnsiString;FL:TStrings;SubDirs:Bool
6563: IncludeItem:TIncludeItemFunc);
6563: Function FileHandlesLeft( MaxHandles : Cardinal ) : Cardinal
6564: Function FileMatchesMask( const FileName, FileMask : AnsiString ) : Boolean
6565: Function FileTimeToStdTime( FileTime : LongInt ) : TStdTimeRec
6566: Function FindNthSlash( const Path : AnsiString; n : Integer ) : Integer
6567: Function FlushOsBuffers( Handle : Integer ) : Boolean
6568: Function GetCurrentUser : AnsiString
6569: Function GetDiskClass( Drive : Char ) : DiskClass
6570: Function GetDiskInfo(Drive:Char;var ClustersAvail,TotalClusters,BytesPerSector, SectorsPerCluster:Cardinal):Bool;
6571: Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Double;var TotalSpaceAvail:Double;var DiskSize:Double):Bool;
6572: Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Comp;var TotalSpaceAvail:Comp;var DiskSize:Comp):Boolean;
6573: { index 0 - FreeBytesAvailable, 1 - TotalNumberOfBytes, 2 - TotalNumberOfFreeBytes }

```

```

6574: Function getDiskSpace2( const path: String; index: integer): int64;
6575: Function GetFileCreateDate( const FileName : AnsiString ) : TDateTime
6576: Function StGetFileLastAccess( const FileName : AnsiString ) : TDateTime
6577: Function GetfileLastModify( const FileName : AnsiString ) : TDateTime
6578: Function GetHomeFolder( aForceSlash : Boolean ) : AnsiString
6579: Function GetLongPath( const APath : AnsiString ) : AnsiString
6580: Function GetMachineName : AnsiString
6581: Function GetMediaID( Drive : Char; var MediaIDRec : MediaIDType ) : Cardinal
6582: Function GetParentFolder( const APath : AnsiString; aForceSlash : Boolean ) : AnsiString
6583: Function GetShortPath( const APath : AnsiString ) : AnsiString
6584: Function GetSystemFolder( aForceSlash : Boolean ) : AnsiString
6585: Function GetTempFolder( aForceSlash : boolean ) : AnsiString
6586: Function StGetWindowsFolder( aForceslash : boolean ) : AnsiString
6587: Function GetWorkingFolder( aForceslash : boolean ) : AnsiString
6588: Function GlobalDateTimeToLocal( const UTC : TStDateTimeRec; MinOffset : Integer ) : TStDateTimeRec
6589: Function StIsDirectory( const DirName : AnsiString ) : Boolean
6590: Function IsDirectoryEmpty( const S : AnsiString ) : Integer
6591: Function IsDriveReady( Drive : Char ) : Boolean
6592: Function IsFile( const FileName : AnsiString ) : Boolean
6593: Function IsFileArchive( const S : AnsiString ) : Integer
6594: Function IsFileHidden( const S : AnsiString ) : Integer
6595: Function IsFileReadOnly( const S : AnsiString ) : Integer
6596: Function IsFileSystem( const S : AnsiString ) : Integer
6597: Function LocalDateTimeToGlobal( const DT1 : TStDateTimeRec; MinOffset : Integer ) : TStDateTimeRec
6598: Function ReadVolumeLabel( var VolName : AnsiString; Drive : Char ) : Cardinal
6599: Function SameFile( const FilePath1, FilePath2 : AnsiString; var ErrorCode : Integer ) : Boolean
6600: Function SetMediaID( Drive : Char; var MediaIDRec : MediaIDType ) : Cardinal
6601: Procedure SplitPath( const APath : AnsiString; Parts : TStrings )
6602: Function StDateTimeToFileTime( const FileTime : TStDateTimeRec ) : LongInt
6603: Function StDateTimeToUnixTime( const DT1 : TStDateTimeRec ) : Longint
6604: Function UnixTimeToStDateTime( UnixTime : Longint ) : TStDateTimeRec
6605: Function ValidDrive( Drive : Char ) : Boolean
6606: Function WriteVolumeLabel( const VolName : AnsiString; Drive : Char ) : Cardinal
6607:
6608: //*****unit uPST_Jc11ANMan;*****
6609: Function CreateAccount( const Server, Username, Fullname, Password, Description, Homedir, Script : string;
  const PasswordNeverExpires : Boolean ) : Boolean
6610: Function CreateLocalAccount( const Username, Fullname, Password, Description, Homedir, Script : string;
  const PasswordNeverExpires : Boolean ) : Boolean
6611: Function DeleteAccount( const Servername, Username : string ) : Boolean
6612: Function DeleteLocalAccount( Username : string ) : Boolean
6613: Function CreateLocalGroup( const Server, Groupname, Description : string ) : Boolean
6614: Function CreateGlobalGroup( const Server, Groupname, Description : string ) : Boolean
6615: Function DeleteLocalGroup( const Server, Groupname : string ) : Boolean
6616: Function GetlocalGroups( const Server : string; const Groups : TStrings ) : Boolean
6617: Function GetGlobalGroups( const Server : string; const Groups : TStrings ) : Boolean
6618: Function LocalGroupExists( const Group : string ) : Boolean
6619: Function GlobalGroupExists( const Server, Group : string ) : Boolean
6620: Function AddAccountToLocalGroup( const Accountname, Groupname : string ) : Boolean
6621: Function LookupGroupName( const Server : string; const RID : TNetWellKnownRID ) : string
6622: Procedure ParseAccountName( const QualifiedName : string; var Domain, UserName : string )
6623: Function IsLocalAccount( const AccountName : string ) : Boolean
6624: Function TimeStampInterval( StartStamp, EndStamp : TDateTime ) : integer
6625: Function GetRandomString( NumChar : cardinal ) : string
6626:
6627: //*****unit uPST_cUtils;*****
6628: TypeS('TUnitType', '( utSrc, utHead, utRes, utPrj, utOther )'
6629: Function cIsWinNT : boolean
6630: Procedure cFilesFromWildcard(Directory,Mask: string;var Files:TStringList;Subdirs,ShowDirs,
  Multitasking:Boolean;
6631: Function cExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer ) : THandle
6632: Function cRunAndGetOutput(Cmd,WorkDir:string; ErrFunc:TErrFunc; LineOutputFunc:TLineOutputFunc;
  CheckAbortFunc : TCheckAbortFunc; ShowReturnValue : Boolean) : string
6633: Function cGetShortName( FileName : string ) : string
6634: Procedure cShowError( Msg : String )
6635: Function cCommaStrToStr( s : string; formatstr : string ) : string
6636: Function cIncludeQuoteIfSpaces( s : string ) : string
6637: Function cIncludeQuoteIfNeeded( s : string ) : string
6638: Procedure cLoadFilefromResource( const FileName : string; ms : TMemoryStream )
6639: Function cValidateFile(const FileName:string; const WorkPath:string;const CheckDirs:boolean):string;
6640: Function cBuildFilter( var value : string; const FLTStyle : TFILTERSET ) : boolean;
6641: Function cBuildFilter1( var value : string; const _filters : array of string ) : boolean;
6642: Function cCodeInstoStr( s : string ) : string
6643: Function cStrtoCodeIns( s : string ) : string
6644: Procedure cStrtoAttr( var Attr : TSynHighlighterAttributes; Value : string )
6645: Function cAttrtoStr( const Attr : TSynHighlighterAttributes ) : string
6646: Procedure cStrtoPoint( var pt : TPoint; value : string )
6647: Function cPointtoStr( const pt : TPoint ) : string
6648: Function cListtoStr( const List : TStrings ) : string
6649: Function ListtoStr( const List : TStrings ) : string
6650: Procedure StrtoList( s : string; const List : TStrings; const delimiter : char )
6651: Procedure cStrtoList( s : string; const List : TStrings; const delimiter : char )
6652: Function cGetFileType( const FileName : string ) : TUnitType
6653: Function cGetExTyp( const FileName : string ) : TExUnitType
6654: Procedure cSetPath( Add : string; const UseOriginal : boolean )
6655: Function cExpandFileto( const FileName : string; const basePath : string ) : string
6656: Function cFileSamePath( const FileName : string; const TestPath : string ) : boolean
6657: Procedure cCloneMenu( const FromMenu : TMenuItem; ToMenu : TMenuItem )
6658: Function cGetLastPos( const SubStr : string; const S : string ) : integer

```

```

6659: Function cGenMakePath( FileName : String ) : String;
6660: Function cGenMakePath2( FileName : String ) : String
6661: Function cGenMakePath1( FileName : String; EscapeSpaces, EncloseInQuotes : Boolean ) : String;
6662: Function cGetRealPath( BrokenFileName : String; Directory : String ) : String
6663: Function cCalcMod( Count : Integer ) : Integer
6664: Function cGetVersionString( FileName : string ) : string
6665: Function cCheckChangeDir( var Dir : string ) : boolean
6666: Function cGetAssociatedProgram( const Extension:string; var Filename,Description: string ):boolean
6667: Function cIsNumeric( s : string ) : boolean
6668: Procedure StrToAttr( var Attr : TSynHighlighterAttributes; Value : string )
6669: Function AttrToStr( const Attr : TSynHighlighterAttributes ) : string
6670: Function GetfileTyp( const FileName : string ) : TUnitType
6671: Function Atoi(const aStr: string): integer
6672: Function Itoa(const aint: integer): string
6673:
6674:
6675: procedure SIRegister_cHTTPUtils(CL: TPPascalCompiler);
6676: begin
6677:   FindClass('TOBJECT'), 'EHTTP
6678:   FindClass('TOBJECT'), 'EHTTPParser
6679:   //AnsiCharSet', 'set of AnsiChar
6680:   AnsiStringArray', 'array of AnsiString
6681:   THTTPProtocolEnum', '( hpNone, hpCustom, hpHTTP, hpHTTPS )
6682:   THTTPVersionEnum', '( hvNone, hvCustom, hvHTTP10, hvHTTP11 )
6683:   THTTPVersion', 'record Version : THTTPVersionEnum; Protocol : TH'
6684:     +'TPPProtocolEnum; CustomProtocol : AnsiString; CustomMajVersion : Integer; '
6685:     +'CustomMinVersion : Integer; end
6686:   THTTPHeaderNameEnum', '( hntCustom, hntHost, hntContentType, hnt'
6687:     +'ContentLength, hntContentTransferEncoding, hntContentLocation, hntContentL'
6688:     +'anguage, hntContentEncoding, hntTransferEncoding, hntDate, hntServer, hntU'
6689:     +'serAgent, hntLocation, hntConnection, hntExpires, hntCacheControl, hntSetC'
6690:     +'ookie, hntCookie, hntAuthorization, hntVia, hntWarning, hntContentRange, h'
6691:     +'ntXForwardedFor, hntPragma, hntXPoweredBy, hntWWWAuthenticate, hntLastModi'
6692:     +'fied, hntETag, hntProxyAuthorization, hntReferer, hntAge, hntAcceptRanges, '
6693:     +' hntAcceptEncoding, hntAcceptLanguage, hntAcceptCharset, hntIfModifiedSince'
6694:     +'e, hntIfUnmodifiedSince, hntRetryAfter, hntUpgrade, hntStatus, hntProxyCon'
6695:     +'nection, hntOrigin, hntKeepAlive )
6696:   THTTPHeaderName', 'record Value : THTTPHeaderNameEnum; Custom: AnsiString; end
6697:   THTTPCustomHeader', 'record FieldName : AnsiString;FieldValue :'
6698:     +' AnsiString; end
6699:   //PHTTPCustomHeader', '^THTTPCustomHeader // will not work
6700:   THTTPContentLengthEnum', '( hcLNone, hcLByteCount )
6701:   THTTPContentLength', 'record Value : THTTPContentLengthEnum; ByteCount:Int64; end
6702:   //PHTTPCustomHeader', '^THTTPCustomHeader // will not work
6703:   THTTPContentTypeMajor', '( hctmCustom, hctmText, hctmImage )
6704:   THTTPContentTypeEnum', '( hctNone, hctCustomParts, hctCustomStri'
6705:     +'ng, hctTextHtml, hctTextAscii, hctTextCss, hctTextPlain, hctTextXml, hctTe'
6706:     +'xtCustom, hctImageJpeg, hctImagePng, hctImageGif, hctImageCustom, hctAppli'
6707:     +'cationJSON, hctApplicationOctetStream, hctApplicationJavaScipt, hctAppli'
6708:     +'cationCustom, hctAudioCustom, hctVideoCustom )
6709:   THTTPContentType', 'record Value : THTTPContentTypeEnum; CustomM'
6710:     +'ajor : AnsiString; CustomMinor : AnsiString; Parameters : AnsiStringArray; '
6711:     +'CustomStr : AnsiString; end
6712:   THTTPDateFieldEnum', '( hdNone, hdCustom, hdParts, hdDateTime )'
6713:   THTTPDateField', 'record Value : THTTPDateFieldEnum; DayOfWeek :'
6714:     +' Integer; Day : integer; Month : integer; Year : Integer; Hour : integer; '
6715:     +'Min : integer; Sec : Integer; TimeZoneGMT : Boolean; CustomTimeZone : Ansi'
6716:     +'String; DateTime : TDateTime; Custom : AnsiString; end
6717:   THTTPTransferEncodingEnum', '( hteNone, hteCustom, hteChunked )'
6718:   THTTPTransferEncoding', 'record Value : THTTPTransferEncodingEnum'
6719:     +'m; Custom : AnsiString; end
6720:   THTTPConnectionFieldEnum', '( hcfnNone, hcfcCustom, hcfcClose, hcfcKeepAlive )'
6721:   THTTPConnectionField', 'record Value : THTTPConnectionFieldEnum; '
6722:     +' Custom : AnsiString; end
6723:   THTTPAgeFieldEnum', '( hafNone, hafCustom, hafAge )'
6724:   THTTPAgeField', 'record Value : THTTPAgeFieldEnum; Age : Int64;Custom:AnsiString; end
6725:   THTTPCacheControlFieldEnum', '( hccfNone, hccfDecoded, hccfCustom )'
6726:   THTTPCacheControlRequestSubField', '( hccsfNoCache, hccsfNoStore'
6727:     +' , hccsfMaxAge, hccsfMaxStale, hccsfMinFresh, hccsfNoTransform, hccsfOnlyIfCached )'
6728:   THTTPCacheControlResponseSubField', '( hccrfPublic, hccrfPrivate'
6729:     +' , hccrfNoCache, hccrfNoStore, hccrfNoTransform, hccrfMustRevalidate, hccrf'
6730:     +'ProxyRevalidate, hccrfMaxAge, hccrfSMaxAge )'
6731:   THTTPCacheControlField', 'record Value : THTTPCacheControlFieldEnum; end
6732:   THTTPContentEncodingEnum', '( hceNone, hceCustom, hceIdentity, h'
6733:     +'ceCompress, hceDeflate, hceExi, hceGzip, hcePack200Gzip )'
6734:   THTTPContentEncoding', 'record Value:THTTPContentEncodingEnum;Custom:AnsiString; end'
6735:   THTTPContentEncodingFieldEnum', '( hcefNone, hcefList )'
6736:   THTTPContentEncodingField', 'record Value : THTTPContentEncoding'
6737:     +'FieldEnum; List : array of THTTPContentEncoding; end
6738:   THTTPREtryAfterFieldEnum', '( hrefNone, hrefCustom, harfDate, harfSeconds )'
6739:   THTTPREtryAfterField', 'record Value : THTTPREtryAfterFieldEnum; '
6740:     +' Custom : AnsiString; Date : TDateTime; Seconds : Int64; end
6741:   THTTPContentRangeFieldEnum', '( hcrfNone, hcrfCustom, hcrfByteRange )'
6742:   THTTPContentRangeField', 'record Value : THTTPContentRangeFieldE'
6743:     +'num; ByteFirst : Int64; ByteLast : Int64; ByteSize : Int64; Custom : AnsiString; end
6744:   THTTPSetCookieFieldEnum', '( hscoNone, hscoDecoded, hscoCustom )'
6745:   THTTPSetCookieCustomField', 'record Name : AnsiString; Value : AnsiString; end'
6746:   THTTPSetCookieCustomFieldArray', 'array of THTTPSetCookieCustomField'
6747:   THTTPSetCookieField', 'record Value : THTTPSetCookieFieldEnum; D'

```

```

6748: +'omain : AnsiString; Path : AnsiString; Expires : THTTPDateField; MaxAge : '
6749: +'Int64; HttpOnly : Boolean; Secure : Boolean; CustomFields : THTTPSetCookie'
6750: +'CustomFieldArray; Custom : AnsiString; end
6751: //THTTPSetCookieField', '^THTTPSetCookieField // will not work
6752: THTTPSetCookieFieldArray', 'array of THTTPSetCookieField
6753: THTTPCookieFieldEnum', '( hcoNone, hcoDecoded, hcoCustom )
6754: THTTPCookieFieldEntry', 'record Name : AnsiString; Value : AnsiString; end
6755: //THTTPCookieFieldEntry', '^THTTPCookieFieldEntry // will not work
6756: THTTPCookieFieldEntryArray', 'array of THTTPCookieFieldEntry
6757: THTTPCookieField', 'record Value : THTTPCookieFieldEnum; Entries'
6758: +' : THTTPCookieFieldEntryArray; Custom : AnsiString; end
6759: THTTPCommonHeaders', 'record TransferEncoding : THTTPTransferEnc'
6760: +'oding; ContentType : THTTPContentType; ContentLength : THTTPContentLength;'
6761: +' Connection : THTTPConnectionField; ProxyConnection : THTTPConnectionField'
6762: +' ; Date : THTTPDateField; ContentEncoding : THTTPContentEncodingField; end
6763: THTTPCustomHeaders', 'array of THTTPCustomHeader
6764: //THTTPFixedHeaders', 'array[THTTPHeaderNameEnum] of AnsiString
6765: THTTPFixedHeaders', 'array[0..42] of AnsiString
6766: THTTPMethodEnum', '( hmNone, hmCustom, hmGET, hmPUT, hmPOST, hmC'
6767: +'ONNECT, hmHEAD, hmDELETE, hmOPTIONS, hmTRACE )'
6768: THTTPMethod', 'record Value : THTTPMethodEnum; Custom : AnsiString; end
6769: THTTPRequestStartLine', 'record Method: THTTPMethod;URI: AnsiString;Version:THTTPVersion; end
6770: THTTPRequestHeader', 'record CommonHeaders : THTTPCommonHeaders;'
6771: +' FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Coo'
6772: +'kie : THTTPCookieField; IfModifiedSince:THTTPDateField;ifUnmodifiedSince:THTTPDateField;end
6773: //THTTPRequestHeader', '^THTTPRequestHeader // will not work
6774: THTTPRequest', 'record StartLine : THTTPRequestStartLine; Header'
6775: +' : THTTPRequestHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6776: THTTPResponseStartLineMessage', '( hslmNone, hslmCustom, hslmOK)'
6777: THTTPResponseStartLine', 'record Version : THTTPVersion; Code : '
6778: +'Integer; Msg : THTTPResponseStartLineMessage; CustomMsg : AnsiString; end
6779: THTTPResponseHeader', 'record CommonHeaders : THTTPCommonHeaders'
6780: +' ; FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Co'
6781: +'okies : THTTPSetCookieFieldArray; Expires : THTTPDateField; LastModified : '
6782: +' THTTPDateField; Age : THTTPAgeField; end
6783: //THTTPResponseHeader', '^THTTPResponseHeader // will not work
6784: THTTPResponse', 'record StartLine : THTTPResponseStartLine; Head'
6785: +'er : THTTPResponseHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6786: Function HTTPMessageHasContent( const H : THTTPCommonHeaders ) : Boolean
6787: Procedure InitHTTPRequest( var A : THTTPRequest )
6788: Procedure InitHTTPResponse( var A : THTTPResponse )
6789: Procedure ClearHTTPVersion( var A : THTTPVersion )
6790: Procedure ClearHTTPContentLength( var A : THTTPContentLength )
6791: Procedure ClearHTTPContentType( var A : THTTPContentType )
6792: Procedure ClearHTTPDateField( var A : THTTPDateField )
6793: Procedure ClearHTTPTransferEncoding( var A : THTTPTransferEncoding )
6794: Procedure ClearHTTPConnectionField( var A : THTTPConnectionField )
6795: Procedure ClearHTTPAgeField( var A : THTTPAgeField )
6796: Procedure ClearHTTPContentEncoding( var A : THTTPContentEncoding )
6797: Procedure ClearHTTPContentEncodingField( var A : THTTPContentEncodingField )
6798: Procedure ClearHTTPContentRangeField( var A : THTTPContentRangeField )
6799: Procedure ClearHTTPSetCookieField( var A : THTTPSetCookieField )
6800: Procedure ClearHTTPCommonHeaders( var A : THTTPCommonHeaders )
6801: //Procedure ClearHTTPFixedHeaders( var A : THTTPFixedHeaders )
6802: Procedure ClearHTTPCustomHeaders( var A : THTTPCustomHeaders )
6803: Procedure ClearHTTPCookieField( var A : THTTPCookieField )
6804: Procedure ClearHTTPMethod( var A : THTTPMethod )
6805: Procedure ClearHTTPRequestStartLine( var A : THTTPRequestStartLine )
6806: Procedure ClearHTTPRequestHeader( var A : THTTPRequestHeader )
6807: Procedure ClearHTTPRequest( var A : THTTPRequest )
6808: Procedure ClearHTTPResponseStartLine( var A : THTTPResponseStartLine )
6809: Procedure ClearHTTPResponseHeader( var A : THTTPResponseHeader )
6810: Procedure ClearHTTPResponse( var A : THTTPResponse )
6811: THTTPStringOption', '( hsoNone )
6812: THTTPStringOptions', 'set of THTTPStringOption
6813: FindClass('TOBJECT'), 'TAnsiStringBuilder
6814:
6815: Procedure BuildStrHTTPVersion(const A:THTTPVersion;const B:TAnsiStringBuilder; P:THTTPStringOptions;
6816: Procedure BuildStrHTTPContentLengthValue(const
A:THTTPContentLength;B:TAnsiStringBuilder;P:THTTPStringOptions)
6817: Procedure BuildStrHTTPContentLength(const A : THTTPContentLength;
B:TAnsiStringBuilder;P:THTTPStringOptions)
6818: Procedure BuildStrHTTPContentTypeValue(const A:THTTPContentType;B:TAnsiStringBuilder;const
P:THTTPStringOptions)
6819: Procedure BuildStrHTTPContentType(const A:THTTPContType;const B:TAnsiStringBuilder; const
P:THTTPStringOptions)
6820: Procedure BuildStrRFCDateTime( const DOW, Da, Mo, Ye, Ho, Mi, Se : Integer; const TZ : AnsiString; const
B : TAnsiStringBuilder; const P : THTTPStringOptions)
6821: Procedure BuildStrHTTPDateFieldValue( const A : THTTPDateField; const B : TAnsiStringBuilder; const P :
THTTPStringOptions)
6822: Procedure BuildStrHTTPDateField(const A:THTTPDateField;const B:TAnsiStringBuilder;const
P:THTTPStringOptions);
6823: Procedure BuildStrHTTPTransferEncodingValue( const A : THTTPTransferEncoding; const B :
TAnsiStringBuilder; const P : THTTPStringOptions)
6824: Procedure BuildStrHTTPTransferEncoding( const A : THTTPTransferEncoding; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6825: Procedure BuildStrHTTPContentRangeField( const A : THTTPContentRangeField; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6826: Procedure BuildStrHTTPConnectionFieldValue( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)

```

```

6827: Procedure BuildStrHTTPConnectionField( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6828: Procedure BuildStrHTTPAgeField(const A:THTTPAgeField;const B:TAnsiStringBuilder;const
P:THTTPStringOptions);
6829: Procedure BuildStrHTTPContentEncoding( const A : THTTPContentEncoding; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6830: Procedure BuildStrHTTPContentEncodingField(const A:THTTPContentEncodingField;const
B:TAnsiStringBuilder;const P:THTTPStringOptions)
6831: Procedure BuildStrHTTPProxyConnectionField(const A : THTTPConnectionField; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6832: Procedure BuildStrHTTPCommonHeaders( const A : THTTPCommonHeaders; const B : TAnsiStringBuilder; const P
: THTTPStringOptions)
6833: Procedure BuildStrHTTPFixedHeaders(const A:THTTPFixedHeaders;const B:TAnsiStrBuilder;const
P:THTTPStringOptions)
6834: Procedure BuildStrHTTPCustomHeaders( const A : THTTPCustomHeaders; const B : TAnsiStringBuilder; const P
: THTTPStringOptions)
6835: Procedure BuildStrHTTPSetCookieFieldValue( const A : THTTPSetCookieField; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6836: Procedure BuildStrHTTPCookieFieldValue( const A : THTTPCookieField; const B : TAnsiStringBuilder; const P
: THTTPStringOptions)
6837: Procedure BuildStrHTTPCookieField(const A:THTTPCookieField;const B:TAnsiStrBuilder;const
P:THTTPStringOptions);
6838: Procedure BuildStrHTTPMethod( const A : THTTPMethod; const B : TAnsiStringBuilder; const P :
THTTPStringOptions)
6839: Procedure BuildStrHTTPRequestStartLine( const A : THTTPRequestStartLine; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6840: Procedure BuildStrHTTPRequestHeader(const A:THTTPRequestHeader;const B:TAnsiStringBuilder;const
P:THTTPStringOptions);
6841: Procedure BuildStrHTTPRequest(const A : THTTPRequest; const B : TAnsiStringBuilder; const P :
THTTPStringOptions)
6842: Procedure BuildStrHTTPResponseCookieFieldValueArray( const A : THTTPSetCookieFieldValueArray; const B
: TAnsiStringBuilder; const P : THTTPStringOptions)
6843: Procedure BuildStrHTTPResponseStartLine(const A:THTTPResponseStartLine;const B:TAnsiStrBldr;const P
THTTPStrOptions);
6844: Procedure BuildStrHTTPResponseHeader(const A:THTTPResHeader;const B:TAnsiStrBuilder;const
P:THTTPStringOptions);
6845: Procedure BuildStrHTTPResponse(const A:THTTPResponse; const B:TAnsiStringBuilder; const
P:THTTPStringOptions);
6846: Function HTTPContentTypeValueToStr( const A : THTTPContentType ) : AnsiString
6847: Function HTTPSetCookieFieldValueToStr( const A : THTTPSetCookieField ) : AnsiString
6848: Function HTTPCookieFieldValueToStr( const A : THTTPCookieField ) : AnsiString
6849: Function HTTPMethodToStr( const A : THTTPMethod ) : AnsiString
6850: Function HTTPRequestToStr( const A : THTTPRequest ) : AnsiString
6851: Function HTTPResponseToStr( const A : THTTPResponse ) : AnsiString
6852: Procedure PrepareCookie(var A:THTTPCookieField;const B:THTTPSetCookieFieldValueArray;const
Domain:AnsiString;const Secure:Bool; THTTPParserHeaderParseFunc', 'Function ( const HeaderName : THTT
6853: +PHeaderNameEnum; const HeaderPtr : __Pointer ) : Boolean
6854: SIRegister_THTTPParser(CL);
6855: FindClass('TOBJECT','THTTPContentDecoder
6856: THTTPContentDecoderProc', 'Procedure ( const Sender : THTTPContentDecoder )
6857: THTTPContentDecoderContentType', '( crctFixedSize, crctChunked, crctUnsized )
6858: THTTPContentDecoderChunkState', '( crcsChunkHeader, crcsContent,
6859: +' crcsContentCRLF, crcsTrailer, crcsFinished )
6860: THTTPContentDecoderLogEvent', 'Procedure ( const Sender : THTTPContentDecoder; const LogMsg : String )
6861: SIRegister_THTTPContentDecoder(CL);
6862: THTTPContentReaderMechanism', '( hcrmEvent, hcrmString, hcrmStream, hcrmFile )
6863: FindClass('TOBJECT','THTTPContentReader
6864: THTTPContentReaderProc', 'Procedure ( const Sender : THTTPContentReader )
6865: THTTPContentReaderLogEvent', 'Procedure(const Sender:THTTPContentReader;const LogMsg:String;const
LogLevel:Int;
6866: SIRegister_THTTPContentReader(CL);
6867: THTTPContentWriterMechanism',(hctmEvent, hctmString, hctmStream, hctmFile )
6868: FindClass('TOBJECT','THTTPContentWriter
6869: THTTPContentWriterLogEvent', 'Procedure ( const Sender : THTTPContentWriter;const LogMsg:AnsiString );
6870: SIRegister_THTTPContentWriter(CL);
6871: Procedure SelfTestcHTTPUtils
6872: end;
6873:
6874: (*-----*)
6875: procedure SIRegister_cTLSUtils(CL: TPPascalCompiler);
6876: begin
6877: 'TLSlibraryVersion', 'String '1.00
6878: 'TLSerror_None', 'LongInt'( 0 );
6879: 'TLSerror_InvalidBuffer', 'LongInt'( 1 );
6880: 'TLSerror_InvalidParameter', 'LongInt'( 2 );
6881: 'TLSerror_InvalidCertificate', 'LongInt'( 3 );
6882: 'TLSerror_InvalidState', 'LongInt'( 4 );
6883: 'TLSerror_DecodeError', 'LongInt'( 5 );
6884: 'TLSerror_BadProtocol', 'LongInt'( 6 );
6885: Function TLSErrorMessage( const TLSerror : Integer ) : String
6886: SIRegister_ETLSError(CL);
6887: TTLSProtocolVersion', 'record major : Byte; minor : Byte; end
6888: TTLSProtocolVersion', '^TTLSProtocolVersion // will not work
6889: Procedure InitSSLProtocolVersion30( var A : TTLSProtocolVersion )
6890: Procedure InitTLSProtocolVersion10( var A : TTLSProtocolVersion )
6891: Procedure InitTLSProtocolVersion11( var A : TTLSProtocolVersion )
6892: Procedure InitTLSProtocolVersion12( var A : TTLSProtocolVersion )
6893: Function IsTLSProtocolVersion( const A, B : TTLSProtocolVersion ) : Boolean
6894: Function IsSSL2( const A : TTLSProtocolVersion ) : Boolean

```

```

6895: Function IsSSL3( const A : TTLSProtocolVersion ) : Boolean
6896: Function IsTLS10( const A : TTLSProtocolVersion ) : Boolean
6897: Function IsTLS11( const A : TTLSProtocolVersion ) : Boolean
6898: Function IsTLS12( const A : TTLSProtocolVersion ) : Boolean
6899: Function IsTLS10OrLater( const A : TTLSProtocolVersion ) : Boolean
6900: Function IsTLS11OrLater( const A : TTLSProtocolVersion ) : Boolean
6901: Function IsTLS12OrLater( const A : TTLSProtocolVersion ) : Boolean
6902: Function IsFutureTLSVersion( const A : TTLSProtocolVersion ) : Boolean
6903: Function IsKnownTLSVersion( const A : TTLSProtocolVersion ) : Boolean
6904: Function TLSProtocolVersionToStr( const A : TTLSProtocolVersion ) : String
6905: Function TLSProtocolVersionName( const A : TTLSProtocolVersion ) : String
6906: PTLSRandom', '^PTLSRandom // will not work
6907: Procedure InitTLSRandom( var Random : PTLSRandom )
6908: Function TLSRandomToStr( const Random : PTLSRandom ) : AnsiString
6909: 'TLSSESSIONIDMaxLen', 'LongInt'( 32 );
6910: Procedure InitTLSSessionID( var SessionID : TTLSSessionID; const A : AnsiString )
6911: Function EncodeTLSSessionID(var Buffer:string;const Size:Int;const SessionID:TTLSSessionID):Int;
6912: Function DecodeTLSSessionID(const Buffer:string;const Size:Int;var SessionID:TTLSSessionID):Int;
6913: TTLSignatureAndHashAlgorithm', 'record Hash : TTLSShHashAlgorithm'
6914: +' ; Signature : TTLSignatureAlgorithm; end
6915: // TTLSignatureAndHashAlgorithm', '^TTLSignatureAndHashAlgorithm +'// will not work
6916: TTLSignatureAndHashAlgorithmArray', 'array of TTLSignatureAndHashAlgorithm
6917: TTLSKeyExchangeAlgorithm', '( tlskeaNone, tlskeaNULL, tlskeaDHE '
6918: +'DSS, tlskeaDHE_RSA, tlskeaDH_Anon, tlskeaRSA, tlskeaDH_DSS, tlskeaDH_RSA )
6919: TTLSMACAlgorithm', '( tlsmNone, tlsmNULL, tlsmAHMAC_MD5, tlsm'
6920: +'HMAC_SHA1, tlsmAHMAC_SHA256, tlsmAHMAC_SHA256, tlsmAHMAC_SHA384, tlsmAHMAC_SHA512 )
6921: TTLSMacAlgorithmInfo', 'record Name : AnsiString; DigestSize : I'
6922: +'nteger; Supported : Boolean; end
6923: PTLSMacAlgorithmInfo', '^TTLSMacAlgorithmInfo // will not work
6924: 'TLS_MAC_MAXDIGESTSIZE', 'LongInt'( 64 );
6925: TTLSPRFAlgorithm', '( tlspaSHA256 )
6926: Function tlsp_MD5( const Secret, Seed : AnsiString; const Size : Integer ) : AnsiString
6927: Function tlsp_SHA1( const Secret, Seed : AnsiString; const Size : Integer ) : AnsiString
6928: Function tlsp_SHA256( const Secret, Seed : AnsiString; const Size : Integer ) : AnsiString
6929: Function tlsp_SHA512( const Secret, Seed : AnsiString; const Size : Integer ) : AnsiString
6930: Function tlsp10PRF( const Secret, ALLabel, Seed : AnsiString; const Size : Integer ) : AnsiString
6931: Function tlsp12PRF_SHA256( const Secret, ALLabel, Seed : AnsiString; const Size : Integer ) : AnsiString
6932: Function tlsp12PRF_SHA512( const Secret, ALLabel, Seed : AnsiString; const Size : Integer ) : AnsiString
6933: Function TLSPRF( const ProtoVersion:TTLSProtocolVersion;const Secret,ALabel,Seed:AString;const
Size:Int):AString;
6934: Function tlsp10KeyBlock( const MasterSecret, ServerRandom, ClientRandom:AnsiString; const
Size:Integer):AnsiString
6935: Function tlsp12SHA256KeyBlock( const MasterSecret, ServerRandom, ClientRandom: AnsiString;const
Size:Int):AnsiString;
6936: Function tlsp12SHA512KeyBlock( const MasterSecret, ServerRandom, ClientRandom: AnsiString;const
Size:Int):AnsiString;
6937: Function TLSKeyBlock( const ProtocolVersion:TTLSProtocolVersion; const MasterSecret, ServerRandom,
ClientRandom : AnsiString; const Size : Integer ) : AnsiString
6938: Function tlsp10MasterSecret( const PreMasterSecret, ClientRandom, ServerRandom:AnsiString ) : AnsiString;
6939: Function tlsp12SHA256MasterSecret( const PreMasterSecret, ClientRandom, ServerRandom:AnsiString ) : AnsiString;
6940: Function tlsp12SHA512MasterSecret( const PreMasterSecret, ClientRandom, ServerRandom:AnsiString ) : AnsiString;
6941: Function TLSMasterSecret( const ProtocolVersion: TTLSProtocolVersion;const PreMasterSecret, ClientRandom,
ServerRandom:AnsiString ) : AnsiString
6942: 'TLSKeys', 'record KeyBlock : AnsiString; ClientMACKey : AnsiStr'
6943: +'ing; ServerMACKey : AnsiString; ClientEncKey : AnsiString; ServerEncKey : '
6944: +'AnsiString; ClientIV : AnsiString; ServerIV : AnsiString; end
6945: Procedure GenerateTLSKeys( const ProtocolVersion : TTLSProtocolVersion; const MACKeyBits, CipherKeyBits,
IVBits : Integer; const MasterSecret, ServerRandom, ClientRandom : AnsiString; var TLSKeys : TTLSKeys )
6946: Procedure GenerateFinalTLSKeys( const ProtocolVersion:TTLSProtocolVersion;const IsExportable:Boolean;const
ExpandedKeyBits:Integer;const ServerRandom,ClientRandom:AnsiString;var TLSKeys:TTLSKeys )
6947: 'TLS_PLAINTEXT_FRAGMENT_MAXSIZE', 'LongInt'( 16384 - 1 );
6948: 'TLS_COMPRESSED_FRAGMENT_MAXSIZE','LongInt'( 16384 + 1024 );
6949: Procedure SelfTestcTLSUtils
6950: end;
6951:
6952: (*-----*)
6953: procedure SIRegister_Reversi(CL: TPSCompiler);
6954: begin
6955:   sPosData', 'record corner : boolean; square2x2 : boolean; edge:boolean; stable : integer; internal :
integer; disks : integer; mx : integer; my : integer; end
6956: // pBoard', '^tBoard // will not work
6957: Function rCalculateData( cc : byte; cx, cy : integer) : sPosData
6958: Function rCheckMove( color : byte; cx, cy : integer) : integer
6959: //Function rDoStep( data : pBoard ) : word
6960: Function winExecAndWait( const sAppPath : string; wVisible : word ) : boolean
6961: end;
6962:
6963: procedure SIRegister_IWDBCommon(CL: TPSCompiler);
6964: begin
6965: Function InEditMode( ADataSet : TDataSet ) : Boolean
6966: Function CheckDataSource( ADataSource : TDataSource ) : Boolean;
6967: Function CheckDataSource1(ADataSource:TDataSource;const AFieldName:string;var VField:TField):boolean;
6968: Function GetFieldText( AField : TField ) : String
6969: end;
6970:
6971: procedure SIRegister_SortGrid(CL: TPSCompiler);
6972: begin
6973:   TPrintMode', '( pmPrint, pmPreview, pmPageCount )
6974:   TMyPrintRange', '( prAll, prSelected )

```

```

6975: TSortStyle', '( ssAutomatic, ssNormal, ssNumeric, ssNumericExten'
6976: +'ded, ssDateTime, ssTime, ssCustom )
6977: TSortDirection', '( sdAscending, sdDescending )
6978: TSetChecked', 'Procedure ( Sender : TObject; ACol, ARow : integer; State : Boolean)
6979: TGetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integer
6980: +'er; var Strs : TStringList; var Width, Height : integer; var Sorted : Boolean)
6981: TSetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integer; Str : String)
6982: TSetEllipsis', 'Procedure ( Sender : TObject; ACol, ARow : integer)
6983: SIRegister_TSortOptions(CL);
6984: SIRegister_TPrintOptions(CL);
6985: TSortedListEntry', 'record Str : String; RowNum : integer; SortOption : TSortOptions; end
6986: SIRegister_TSortedList(CL);
6987: TCellBevelStyle', '( cbNone, cbRaised, cbLowered )
6988: TCellBevel', 'record Style: TCellBevelStyle; UpperLeftColor: TColor; LowerRightColor : TColor; end
6989: TVertAlignment', '( taTopJustify, taBottomJustify, taMiddle )
6990: TFormatOptions', 'record Brush : TBrush; Font : TFont; Alignment'
6991: +'Horz : TAlignment; AlignmentVert : TVertAlignment; Bevel : TCellBevel; HideText : Boolean; end
6992: SIRegister_TFontSetting(CL);
6993: SIRegister_TFontList(CL);
6994: AddTypeS(TFormatDrawCellEvent', 'Procedure ( Sender : TObject; Col, Row : '
6995: +'integer; State: TGridDrawState; var FormatOptions: TFormatOptions; var CheckBox, Combobox, Ellipsis: Boolean);
6996: TSetFilterEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
6997: TSearchEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
6998: TUpdateGridEvent', 'Procedure ( Sender : TObject; ARow : integer)
6999: TSizeChangedEvent', 'Procedure ( Sender : TObject; OldColCount, OldRowCount : integer)
7000: TBeginSortEvent', 'Procedure ( Sender : TObject; var Col : integer)
7001: TEndSortEvent', 'Procedure ( Sender : TObject; Col : integer)
7002: TGetSortStyleEvent', 'Procedure ( Sender : TObject; Col : integer'
7003: +'r; var SortStyle : TSortStyle)
7004: TCellValidateEvent', 'Procedure ( Sender : TObject; ACol, ARow : '
7005: +'integer; const OldValue : string; var NewValue : String; var Valid : Boolean)
7006: SIRegister_TSortGrid(CL);
7007: Function ExtendedCompare( const Str1, Str2 : String) : Integer
7008: Function NormalCompare( const Str1, Str2 : String) : Integer
7009: Function DateTimeCompare( const Str1, Str2 : String) : Integer
7010: Function NumericCompare( const Str1, Str2 : String) : Integer
7011: Function TimeCompare( const Str1, Str2 : String) : Integer
7012: //Function Compare( Item1, Item2 : Pointer) : Integer
7013: end;
7014:
7015: ***** procedure Register_IB(CL: TPSPascalCompiler);
7016: Procedure IBAalloc( var P, OldSize, NewSize : Integer)
7017: Procedure IBEror( ErrMess : TIBClientError; const Args : array of const)
7018: Procedure IBDataBaseError
7019: Function StatusVector : PISC_STATUS
7020: Function StatusVectorArray : PStatusVector
7021: Function CheckStatusVector( ErrorCode : array of ISC_STATUS) : Boolean
7022: Function StatusVectorAsText : string
7023: Procedure SetIBDataBaseErrorMessages( Value : TIBDataBaseErrorMessages)
7024: Function GetIBDataBaseErrorMessages : TIBDataBaseErrorMessages
7025:
7026:
7027: //*****unit uPSI_BoldUtils;*****
7028: Function CharCount( c : char; const s : string) : integer
7029: Function BoldNamesEqual( const name1, name2 : string) : Boolean
7030: Procedure BoldAppendToStrings(strings: TStringList; const ForceNewLine:Boolean)
7031: Function BoldSeparateStringList(strings: TStringList; const Separator, PreString, PostString:String):String
7032: Function BoldSeparatedAppend( const S1, S2 : string; const Separator : string) : string
7033: Function BoldTrim( const S : string) : string
7034: Function BoldIsPrefix( const S, Prefix : string) : Boolean
7035: Function BoldStrEqual( P1, P2 : PChar; Len : integer) : Boolean
7036: Function BoldStrAnsiEqual( P1, P2 : PChar; Len : integer) : Boolean
7037: Function BoldAnsiEqual( const S1, S2 : string) : Boolean
7038: Function BoldStrStringEqual( const S1 : string; P2 : PChar; Len : integer) : Boolean
7039: Function BoldCaseIndependentPos( const Substr, S : string) : Integer
7040: //Procedure EnumToStrings( aTypeInfo : pTypeInfo; Strings : TStrings)
7041: Function CapitalisedToSpaced( Capitalised : String) : String
7042: Function SpacedToCapitalised( Spaced : String) : String
7043: Function BooleanToString( BoolValue : Boolean) : String
7044: Function StringToBoolean( StrValue : String) : Boolean
7045: Function GetUpperLimitForMultiplicity( const Multiplicity : String) : Integer
7046: Function GetLowerLimitForMultiplicity( const Multiplicity : String) : Integer
7047: Function StringListToVarArray( List : TStringList) : variant
7048: Function IsLocalMachine( const Machinename : WideString) : Boolean
7049: Function GetComputerNameStr : string
7050: Function TimeStampComp( const Timel, Time2 : TTimeStamp) : Integer
7051: Function BoldStrToDateFmt(const S:string;const DateFormat:string;const DateSeparatorChar:char):TDateTime
7052: Function BoldDateToStrFmt(const aDate:TDateTime;DateFormat:string;const DateSeparatorChar:char):String;
7053: Function BoldParseFormattedDateList(const value:String;const formats:TStrings;var Date:TDateTime):Boolean;
7054: Function BoldParseFormattedDate(const value:String;const formats:array of string; var Date:TDateTime):Boolean;
7055: Procedure EnsureTrailing( var Str : String; ch : char)
7056: Function BoldDirectoryExists( const Name : string) : Boolean
7057: Function BoldForceDirectories( Dir : string) : Boolean
7058: Function BoldRootRegistryKey : string
7059: Function GetModuleFileNameAsString( IncludePath : Boolean) : string
7060: Function BoldVariantToStrings( V : OleVariant; Strings : TStrings) : Integer
7061: Function LogicalAnd( A, B : Integer) : Boolean
7062: record TByHandleFileInformation dwFileAttributes : DWORD;

```

```

7063: +'ftCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime '
7064: +' : TFileTime; dwVolumeSerialNumber : DWORD; nFileSizeHigh : DWORD; nFileSize'
7065: +'eLow : DWORD; nNumberOfLinks : DWORD; nFileIndexHigh : DWORD; nFileIndexLow : DWORD; end
7066: Function GetfileInformationByHandle(hFile:THandle;var lpfileInformation:TByHandlefileInformation):BOOL;
7067: Function IsFirstInstance : Boolean
7068: Procedure ActivateFirst( AString : PChar)
7069: Procedure ActivateFirstCommandLine
7070: function MakeAckPkt(const BlockNumber: Word): string;
7071: procedure SendError(UDPBase:TIdUDPBase;APeerIP:string;const APort:Int;const ErrNumber:Word;ErrStr:string);
7072: procedure SendError(UDPClient: TIdUDPClient; const ErrNumber: Word; ErrorString: string); overload;
7073: procedure SendError(UDPBase: TIdUDPBase; APeerIP: string; const APort: Integer; E: Exception); overload;
7074: procedure SendError(UDPClient: TIdUDPClient; E: Exception); overload;
7075: function IdStrToWord(const Value: string): Word;
7076: function IdWordToStr(const Value: Word): WordStr;
7077: Function HasInstructionSet( const InstructionSet : TCPUIInstructionSet ) : Boolean
7078: Function CPUFeatures : TCPUFeatures
7079:
7080: procedure SIRegister_xrtl_util_CPUUtils(CL: TPPSPascalCompiler);
7081: begin
7082:   AddTypeS('TXRTLBIndex', 'Integer'
7083:   Function XRTLSwapBits( Data : Cardinal; Bit1Index, Bit2Index : TXRTLBIndex ) : Cardinal
7084:   Function XRTLBTest( Data : Cardinal; BitIndex : TXRTLBIndex ) : Boolean
7085:   Function XRTLBSet( Data : Cardinal; BitIndex : TXRTLBIndex ) : Cardinal
7086:   Function XRTLBReset( Data : Cardinal; BitIndex : TXRTLBIndex ) : Cardinal
7087:   Function XRTLBComplement( Data : Cardinal; BitIndex : TXRTLBIndex ) : Cardinal
7088:   Function XRTLSSwapHiLo16( X : Word ) : Word
7089:   Function XRTLSSwapHiLo32( X : Cardinal ) : Cardinal
7090:   Function XRTLSSwapHiLo64( X : Int64 ) : Int64
7091:   Function XRTLROL32( A, S : Cardinal ) : Cardinal
7092:   Function XRTLROLR32( A, S : Cardinal ) : Cardinal
7093:   Function XRTLROL16( A : Word; S : Cardinal ) : Word
7094:   Function XRTLROL16( A : Word; S : Cardinal ) : Word
7095:   Function XRTLROL8( A : Byte; S : Cardinal ) : Byte
7096:   Function XRTLROL8( A : Byte; S : Cardinal ) : Byte
7097: //Procedure XRTLXorBlock( I1, I2, O1 : PByteArray; Len : integer)
7098: //Procedure XRTLIncBlock( P : PByteArray; Len : integer)
7099: Procedure XRTLUMul64( const A, B : Integer; var MulL, MulH : Integer )
7100: Function XRTLPopulation( A : Cardinal ) : Cardinal
7101: end;
7102:
7103: Function XRTLURLDecode( const ASrc : WideString ) : WideString
7104: Function XRTLURLEncode( const ASrc : WideString ) : WideString
7105: Function XRTLURINormalize( const AURI : WideString ) : WideString
7106: Procedure XRTLURIParse(const AURI:WideString;var VProtocol,VHost,VPath,VDocument,VPort,VBookmark,VUserName,
    VPassword : WideString)
7107: Function XRTLEExtractLongPathName(APath: string): string;
7108:
7109: procedure SIRegister_cFundamentUtils(CL: TPPSPascalCompiler);
7110: begin
7111:   AddTypeS('Int8', 'ShortInt
7112:   AddTypeS('Int16', 'SmallInt
7113:   AddTypeS('Int32', 'LongInt
7114:   AddTypeS('UInt8', 'Byte
7115:   AddTypeS('UInt16', 'Word
7116:   AddTypeS('UInt32', 'LongWord
7117:   AddTypeS('UInt64', 'Int64
7118:   AddTypeS('Word8', 'UInt8
7119:   AddTypeS('Word16', 'UInt16
7120:   AddTypeS('Word32', 'UInt32
7121:   AddTypeS('Word64', 'UInt64
7122:   AddTypeS('LargeInt', 'Int64
7123:   AddTypeS('NativeInt', 'Integer
7124:   AddTypeS('NativeUInt', 'Cardinal
7125:   Const('BitsPerByte','LongInt'( 8 );
7126:   Const('BitsPerWord','LongInt'( 16 );
7127:   Const('BitsPerLongWord','LongInt'( 32 );
7128: //Const('BitsPerCardinal','LongInt'( BytesPerCardinal * 8 );
7129: //Const('BitsPerNativeWord','LongInt'( BytesPerNativeWord * 8 );
7130: Function MinI( const A, B : Integer ) : Integer
7131: Function MaxI( const A, B : Integer ) : Integer
7132: Function MinC( const A, B : Cardinal ) : Cardinal
7133: Function MaxC( const A, B : Cardinal ) : Cardinal
7134: Function SumClipI( const A, I : Integer ) : Integer
7135: Function SumClipC( const A : Cardinal; const I : Integer ) : Cardinal
7136: Function InByteRange( const A : Int64 ) : Boolean
7137: Function InWordRange( const A : Int64 ) : Boolean
7138: Function InLongWordRange( const A : Int64 ) : Boolean
7139: Function InShortIntRange( const A : Int64 ) : Boolean
7140: Function InSmallIntRange( const A : Int64 ) : Boolean
7141: Function InLongIntRange( const A : Int64 ) : Boolean
7142: AddTypeS('Bool8', 'ByteBool
7143: AddTypeS('Bool16', 'WordBool
7144: AddTypeS('Bool32', 'LongBool
7145: AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )
7146: AddTypeS('TCompareResultSet', 'set of TCompareResult
7147: Function ReverseCompareResult( const C : TCompareResult ) : TCompareResult
7148: Const('MinSingle','Single').setExtended( 1.5E-45 );
7149: Const('MaxSingle','Single').setExtended( 3.4E+38 );
7150: Const('MinDouble','Double').setExtended( 5.0E-324 );

```

```

7151: Const('MaxDouble','Double').setExtended( 1.7E+308);
7152: Const('MinExtended','Extended').setExtended(3.4E-4932);
7153: Const('MaxExtended','Extended').setExtended(1.1E+4932);
7154: Const('MinCurrency','Currency').SetExtended( - 922337203685477.5807);
7155: Const('MaxCurrency','Currency').SetExtended( 922337203685477.5807);
7156: Function MinF( const A, B : Float ) : Float
7157: Function MaxF( const A, B : Float ) : Float
7158: Function ClipF( const Value : Float; const Low, High : Float ) : Float
7159: Function InSingleRange( const A : Float ) : Boolean
7160: Function InDoubleRange( const A : Float ) : Boolean
7161: Function InCurrencyRange( const A : Float ) : Boolean;
7162: Function InCurrencyRangel( const A : Int64 ) : Boolean;
7163: Function FloatExponentBase2( const A : Extended; var Exponent : Integer ) : Boolean
7164: Function FloatExponentBase10( const A : Extended; var Exponent : Integer ) : Boolean
7165: Function FloatIsInfinity( const A : Extended ) : Boolean
7166: Function FloatIsNaN( const A : Extended ) : Boolean
7167: Const('SingleCompareDelta','Extended').setExtended( 1.0E-34);
7168: Const('DoubleCompareDelta','Extended').setExtended( 1.0E-280);
7169: Const('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
7170: Const('DefaultCompareDelta','Extended').SetExtended( 1.0E-34);
7171: Function FloatZero( const A : Float; const CompareDelta : Float ) : Boolean
7172: Function FloatOne( const A : Float; const CompareDelta : Float ) : Boolean
7173: Function FloatsEqual( const A, B : Float; const CompareDelta : Float ) : Boolean
7174: Function FloatsCompare( const A, B : Float; const CompareDelta : Float ) : TCompareResult
7175: Const('SingleCompareEpsilon','Extended').setExtended( 1.0E-5);
7176: Const('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13);
7177: Const('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17);
7178: Const('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
7179: Function ApproxEqual( const A, B : Extended; const CompareEpsilon : Double ) : Boolean
7180: Function ApproxCompare( const A, B : Extended; const CompareEpsilon : Double) : TCompareResult
7181: Function cClearBit( const Value, BitIndex : LongWord ) : LongWord
7182: Function cSetBit( const Value, BitIndex : LongWord ) : LongWord
7183: Function cIsBitSet( const Value, BitIndex : LongWord ) : Boolean
7184: Function cToggleBit( const Value, BitIndex : LongWord ) : LongWord
7185: Function cISHighBitSet( const Value : LongWord ) : Boolean
7186: Function SetBitScanForward( const Value : LongWord ) : Integer;
7187: Function SetBitScanForward1( const Value, BitIndex : LongWord ) : Integer;
7188: Function SetBitScanReverse( const Value : LongWord ) : Integer;
7189: Function SetBitScanReverse1( const Value, BitIndex : LongWord ) : Integer;
7190: Function ClearBitScanForward( const Value : LongWord ) : Integer;
7191: Function ClearBitScanForward1( const Value, BitIndex : LongWord ) : Integer;
7192: Function ClearBitScanReverse( const Value : LongWord ) : Integer;
7193: Function ClearBitScanReverse1( const Value, BitIndex : LongWord ) : Integer;
7194: Function cReverseBits( const Value : LongWord ) : LongWord;
7195: Function cReversebits1( const Value : LongWord; const BitCount : Integer ) : LongWord;
7196: Function cSwapEndian( const Value : LongWord ) : LongWord
7197: Function cTwosComplement( const Value : LongWord ) : LongWord
7198: Function RotateLeftBits16( const Value : Word; const Bits : Byte ) : Word
7199: Function RotateLeftBits32( const Value : LongWord; const Bits : Byte ) : LongWord
7200: Function RotateRightBits16( const Value : Word; const Bits : Byte ) : Word
7201: Function RotateRightBits32( const Value : LongWord; const Bits : Byte ) : LongWord
7202: Function cBitCount( const Value : LongWord ) : LongWord
7203: Function cIsPowerOfTwo( const Value : LongWord ) : Boolean
7204: Function LowbitMask( const HighBitIndex : LongWord ) : LongWord
7205: Function HighBitMask( const LowBitIndex : LongWord ) : LongWord
7206: Function RangeBitMask( const LowBitIndex, HighBitIndex : LongWord ) : LongWord
7207: Function SetBitRange( const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord ) : LongWord
7208: Function ClearBitRange(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7209: Function ToggleBitRange(const Value:LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7210: Function IsBitRangeSet(const Value: LongWord; const LowBitIndex,HighBitIndex : LongWord) : Boolean
7211: Function IsBitRangeClear(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord): Boolean
7212: // AddTypeS(' CharSet', 'set of AnsiChar'
7213: AddTypeS(' CharSet', 'set of Char' //!!!
7214: AddTypeS(' AnsiCharSet', 'TCharSet'
7215: AddTypeS(' ByteSet', 'set of Byte'
7216: AddTypeS(' AnsiChar', 'Char'
7217: // Function AsCharSet( const C : array of AnsiChar ) : CharSet
7218: Function AsByteSet( const C : array of Byte ) : ByteSet
7219: Procedure ComplementChar( var C : CharSet; const Ch : Char )
7220: Procedure ClearCharSet( var C : CharSet)
7221: Procedure FillCharSet( var C : CharSet)
7222: Procedure ComplementCharSet( var C : CharSet)
7223: Procedure AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7224: Procedure Union( var DestSet : CharSet; const SourceSet : CharSet)
7225: Procedure Difference( var DestSet : CharSet; const SourceSet : CharSet)
7226: Procedure Intersection( var DestSet : CharSet; const SourceSet : CharSet)
7227: Procedure XORCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7228: Function IsSubSet( const A, B : CharSet ) : Boolean
7229: Function IsEqual( const A, B : CharSet ) : Boolean
7230: Function IsEmpty( const C : CharSet ) : Boolean
7231: Function IsComplete( const C : CharSet ) : Boolean
7232: Function cCharCount( const C : CharSet ) : Integer
7233: Procedure ConvertCaseInsensitive( var C : CharSet)
7234: Function CaseInsensitiveCharSet( const C : CharSet ) : CharSet
7235: Function IntRangeLength( const Low, High : Integer ) : Int64
7236: Function IntRangeAdjacent( const Low1, High1, Low2, High2 : Integer ) : Boolean
7237: Function IntRangeOverlap( const Low1, High1, Low2, High2 : Integer ) : Boolean
7238: Function IntRangeHasElement( const Low, High, Element : Integer ) : Boolean
7239: Function IntRangeIncludeElement( var Low, High : Integer; const Element : Integer ) : Boolean

```

```

7240: Function IntRangeIncludeElementRange(var Low, High: Integer; const LowElement, HighElement: Integer): Boolean
7241: Function CardRangeLength( const Low, High : Cardinal ) : Int64
7242: Function CardRangeAdjacent( const Low1, High1, Low2, High2 : Cardinal ) : Boolean
7243: Function CardRangeOverlap( const Low1, High1, Low2, High2 : Cardinal ) : Boolean
7244: Function CardRangeHasElement( const Low, High, Element : Cardinal ) : Boolean
7245: Function CardRangeIncludeElement( var Low, High : Cardinal; const Element : Cardinal ) : Boolean
7246: Function CardRangeIncludeElementRange(var Low, High:Card;const LowElement,HighElement:Card):Boolean;
7247: AddTypeS('UnicodeChar', 'WideChar'
7248: Function Compare( const I1, I2 : Boolean ) : TCompareResult;
7249: Function CompareI( const I1, I2 : Integer ) : TCompareResult;
7250: Function Compare2( const I1, I2 : Int64 ) : TCompareResult;
7251: Function Compare3( const I1, I2 : Extended ) : TCompareResult;
7252: Function CompareA( const I1, I2 : AnsiString ) : TCompareResult
7253: Function CompareW( const I1, I2 : WideString ) : TCompareResult
7254: Function cSgn( const A : LongInt ) : Integer;
7255: Function cSgn1( const A : Int64 ) : Integer;
7256: Function cSgn2( const A : Extended ) : Integer;
7257: AddTypeS('TConvertResult', '( convertOK, convertFormatError, convertOverflow )
7258: Function AnsiCharToInt( const A : AnsiChar ) : Integer
7259: Function WideCharToInt( const A : WideChar ) : Integer
7260: Function CharToInt( const A : Char ) : Integer
7261: Function IntToAnsiChar( const A : Integer ) : AnsiChar
7262: Function IntToWideChar( const A : Integer ) : WideChar
7263: Function IntToChar( const A : Integer ) : Char
7264: Function IsHexAnsiChar( const Ch : AnsiChar ) : Boolean
7265: Function IsHexWideChar( const Ch : WideChar ) : Boolean
7266: Function IsHexChar( const Ch : Char ) : Boolean
7267: Function HexAnsiCharToInt( const A : AnsiChar ) : Integer
7268: Function HexWideCharToInt( const A : WideChar ) : Integer
7269: Function HexCharToInt( const A : Char ) : Integer
7270: Function IntToUpperHexAnsiChar( const A : Integer ) : AnsiChar
7271: Function IntToUpperHexWideChar( const A : Integer ) : WideChar
7272: Function IntToUpperHexChar( const A : Integer ) : Char
7273: Function IntToLowerHexAnsiChar( const A : Integer ) : AnsiChar
7274: Function IntToLowerHexWideChar( const A : Integer ) : WideChar
7275: Function IntToLowerHexChar( const A : Integer ) : Char
7276: Function IntToStringA( const A : Int64 ) : AnsiString
7277: Function IntToStringW( const A : Int64 ) : WideString
7278: Function IntToString( const A : Int64 ) : String
7279: Function UIntToStringA( const A : NativeUInt ) : AnsiString
7280: Function UIntToStringW( const A : NativeUInt ) : WideString
7281: Function UIntToString( const A : NativeUInt ) : String
7282: Function LongWordToStrA( const A : LongWord; const Digits : Integer ) : AnsiString
7283: Function LongWordToStrW( const A : LongWord; const Digits : Integer ) : WideString
7284: Function LongWordToStrU( const A : LongWord; const Digits : Integer ) : UnicodeString
7285: Function LongWordToStr( const A : LongWord; const Digits : Integer ) : String
7286: Function LongWordToHexA(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):AnsiString;
7287: Function LongWordToHexW(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):WideString;
7288: Function LongWordToHex( const A : LongWord; const Digits : Integer;const UpperCase:Boolean):String
7289: Function LongWordToOctA( const A : LongWord; const Digits : Integer ) : AnsiString
7290: Function LongWordToOctW( const A : LongWord; const Digits : Integer ) : WideString
7291: Function LongWordToOct( const A : LongWord; const Digits : Integer ) : String
7292: Function LongWordToBinA( const A : LongWord; const Digits : Integer ) : AnsiString
7293: Function LongWordToBinW( const A : LongWord; const Digits : Integer ) : WideString
7294: Function LongWordToBin( const A : LongWord; const Digits : Integer ) : String
7295: Function TryStringToInt64A( const S : AnsiString; out A : Int64 ) : Boolean
7296: Function TryStringToInt64W( const S : WideString; out A : Int64 ) : Boolean
7297: Function TryStringToInt64( const S : String; out A : Int64 ) : Boolean
7298: Function StringToInt64DefA( const S : AnsiString; const Default : Int64 ) : Int64
7299: Function StringToInt64DefW( const S : WideString; const Default : Int64 ) : Int64
7300: Function StringToInt64Def( const S : String; const Default : Int64 ) : Int64
7301: Function StringToInt64A( const S : AnsiString ) : Int64
7302: Function StringToInt64W( const S : WideString ) : Int64
7303: Function StringToInt64( const S : String ) : Int64
7304: Function TryStringToIntA( const S : AnsiString; out A : Integer ) : Boolean
7305: Function TryStringToIntW( const S : WideString; out A : Integer ) : Boolean
7306: Function TryStringToInt( const S : String; out A : Integer ) : Boolean
7307: Function StringToIntDefA( const S : AnsiString; const Default : Integer ) : Integer
7308: Function StringToIntDefW( const S : WideString; const Default : Integer ) : Integer
7309: Function StringToIntDef( const S : String; const Default : Integer ) : Integer
7310: Function StringToIntA( const S : AnsiString ) : Integer
7311: Function StringToIntW( const S : WideString ) : Integer
7312: Function StringToInt( const S : String ) : Integer
7313: Function TryStringToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7314: Function TryStringToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7315: Function TryStringToLongWord( const S : String; out A : LongWord ) : Boolean
7316: Function StringToLongWordA( const S : AnsiString ) : LongWord
7317: Function StringToLongWordW( const S : WideString ) : LongWord
7318: Function StringToLongWord( const S : String ) : LongWord
7319: Function HexToUIntA( const S : AnsiString ) : NativeUInt
7320: Function HexToUIntW( const S : WideString ) : NativeUInt
7321: Function HexToInt( const S : String ) : NativeUInt
7322: Function TryHexToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7323: Function TryHexToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7324: Function TryHexToLongWord( const S : String; out A : LongWord ) : Boolean
7325: Function HexToLongWordA( const S : AnsiString ) : LongWord
7326: Function HexToLongWordW( const S : WideString ) : LongWord
7327: Function HexToLongWord( const S : String ) : LongWord
7328: Function TryOctToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean

```

```

7329: Function TryOctToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7330: Function TryOctToLongWord( const S : String; out A : LongWord ) : Boolean
7331: Function OctToLongWordA( const S : AnsiString ) : LongWord
7332: Function OctToLongWordW( const S : WideString ) : LongWord
7333: Function OctToLongWord( const S : String ) : LongWord
7334: Function TryBinToLongWordA( const S : AnsiString; out A : LongWord ) : Boolean
7335: Function TryBinToLongWordW( const S : WideString; out A : LongWord ) : Boolean
7336: Function TryBinToLongWord( const S : String; out A : LongWord ) : Boolean
7337: Function BinToLongWordA( const S : AnsiString ) : LongWord
7338: Function BinToLongWordW( const S : WideString ) : LongWord
7339: Function BinToLongWord( const S : String ) : LongWord
7340: Function FloatToStringA( const A : Extended ) : AnsiString
7341: Function FloatToStringW( const A : Extended ) : WideString
7342: Function FloatToString( const A : Extended ) : String
7343: Function TryStringToFloatA( const A : AnsiString; out B : Extended ) : Boolean
7344: Function TryStringToFloatW( const A : WideString; out B : Extended ) : Boolean
7345: Function TryStringToFloat( const A : String; out B : Extended ) : Boolean
7346: Function StringToFloatA( const A : AnsiString ) : Extended
7347: Function StringToFloatW( const A : WideString ) : Extended
7348: Function StringToFloat( const A : String ) : Extended
7349: Function StringToFloatA( const A : AnsiString; const Default : Extended ) : Extended
7350: Function StringToFloatDefW( const A : WideString; const Default : Extended ) : Extended
7351: Function StringToFloatDef( const A : String; const Default : Extended ) : Extended
7352: Function EncodeBase64( const S:Alphabet:AnsiString; const Pad:Boolean;const PadMultiple:Integer;const PadChar: AnsiChar ) : AnsiString
7353: Function DecodeBase64( const S, Alphabet : AnsiString; const PadSet : CharSet ) : AnsiString
7354: unit uPSI_cfundamentUtils;
7355: Const('b64_MIMEBase64','Str').String('ABCDEFIGHJKLMNOPQRSTUVWXYZabcdefghi jklmnopqrstuvwxyz0123456789+/'
7356: Const('b64_UUEncode','String').String('!'#$%&'')*+,.-./0123456789:;<=>?@ABCDEFGHIJKLM NOPQRSTUVWXYZ[\]^_';
7357: Const('b64_XXEncode','String').String('+0123456789ABCDEFGHIJKLM NOPQRSTUVWXYZabcdefghi jklmnopqrstuvwxyz';
7358: Const('CCHARSET','String>b64_XXEncode');
7359: Const('CHEXSET','String'0123456789ABCDEF
7360: Const('HEXDIGITS','String'0123456789ABCDEF
7361: StHexDigits : array[0..$F] of Char = '0123456789ABCDEF';
7362: Const('DIGISET','String'0123456789
7363: Const('LETTERSET','String'ABCDEFGHIJKLM NOPQRSTUVWXYZ'
7364: Const('DIGISET2','TCharset').SetSet('0123456789'
7365: Const('LETTERSET2','TCharset').SetSet('ABCDEFGHIJKLM NOPQRSTUVWXYZ'
7366: Const('HEXSET2','TCharset').SetSet('0123456789ABCDEF');
7367: Const('NUMBERSET','TCharset').SetSet('0123456789');
7368: Const('NUMBERS','String'0123456789');
7369: Const('LETTERS','String'ABCDEFGHIJKLM NOPQRSTUVWXYZ');
7370: Function CharSetToStr( const C : CharSet ) : AnsiString
7371: Function StrToCharSet( const S : AnsiString ) : CharSet
7372: Function MIMEBase64Decode( const S : AnsiString ) : AnsiString
7373: Function MIMEBase64Encode( const S : AnsiString ) : AnsiString
7374: Function UUDecode( const S : AnsiString ) : AnsiString
7375: Function XXDecode( const S : AnsiString ) : AnsiString
7376: Function BytesToHex( const P : array of Byte; const UpperCase : Boolean ) : AnsiString
7377: Function InterfaceToStrA( const I : IInterface ) : AnsiString
7378: Function InterfaceToStrW( const I : IInterface ) : WideString
7379: Function InterfaceToStr( const I : IInterface ) : String
7380: Function ObjectClassName( const O : TObject ) : String
7381: Function ClassClassName( const C : TClass ) : String
7382: Function ObjectToStr( const O : TObject ) : String
7383: Function ObjectToString( const O : TObject ) : String
7384: Function CharSetToStr( const C : CharSet ) : AnsiString
7385: Function StrToCharSet( const S : AnsiString ) : CharSet
7386: Function HashStrA( const S : AnsiString; const Index : Integer; const Count : Integer; const AsciiCaseSensitive : Boolean; const Slots : LongWord ) : LongWord
7387: Function HashStrW( const S:WideString;const Index:Integer;const Count:Integer;const AsciiCaseSensitive:Boolean; const Slots:LongWord ) : LongWord
7388: Function HashStr( const S : String; const Index : Integer; const Count : Integer; const AsciiCaseSensitive : Boolean; const Slots : LongWord ) : LongWord
7389: Function HashInteger( const I : Integer; const Slots : LongWord ) : LongWord
7390: Function HashLongWord( const I : LongWord; const Slots : LongWord ) : LongWord
7391: Const('Bytes1KB','LongInt'( 1024));
7392: SIRegister_IInterface(CL);
7393: Procedure SelfTestCFundamentUtils
7394:
7395: Function CreateSchedule : IJclSchedule
7396: Function NullStamp : TTimeStamp
7397: Function CompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp ) : Int64
7398: Function EqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp ) : Boolean
7399: Function IsNullTimeStamp( const Stamp : TTimeStamp ) : Boolean
7400:
7401: procedure SIRegister_uwinplot(CL: TPSPascalCompiler);
7402: begin
7403: AddTypeS('TFunc', 'function(X : Float) : Float;
7404: Function InitGraphics( Width, Height : Integer ) : Boolean
7405: Procedure SetWindow( Canvas : TCanvas; X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean )
7406: Procedure SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float )
7407: Procedure SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float )
7408: Procedure GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float )
7409: Procedure GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float )
7410: Procedure SetGraphTitle( Title : String )
7411: Procedure SetOxTitle( Title : String )
7412: Procedure SetOyTitle( Title : String )
7413: Function GetGraphTitle : String

```

```

7414: Function GetOxTitle : String
7415: Function GetOyTitle : String
7416: Procedure PlotOxAxis( Canvas : TCanvas)
7417: Procedure PlotOyAxis( Canvas : TCanvas)
7418: Procedure PlotGrid( Canvas : TCanvas; Grid : TGrid)
7419: Procedure WriteGraphTitle( Canvas : TCanvas)
7420: Function SetMaxCurv( NCurv : Byte) : Boolean
7421: Procedure SetPointParam( CurvIndex, Symbol, Size : Integer; Color : TColor)
7422: Procedure SetLineParam( CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor)
7423: Procedure SetCurvLegend( CurvIndex : Integer; Legend : String)
7424: Procedure SetCurvStep( CurvIndex, Step : Integer)
7425: Function GetMaxCurv : Byte
7426: Procedure GetPointParam( CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor)
7427: Procedure GetLineParam(CurvIndex:Integer;var Style:TPenStyle;var Width:Integer; var Color:TColor);
7428: Function GetCurvLegend( CurvIndex : Integer) : String
7429: Function GetCurvStep( CurvIndex : Integer) : Integer
7430: Procedure PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Integer)
7431: Procedure PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Integer)
7432: Procedure PlotCurveWithErrorBars(Canvas : TCanvas; X,Y,S: TVector; Ns,Lb,Ub,CurvIndex:Integer)
7433: Procedure PlotFunc(Canvas: TCanvas; Func: TFunc; Xmin,Xmax: Float; Npt,CurvIndex: Integer)
7434: Procedure WriteLegend( Canvas : TCanvas; NCurv : Integer; ShowPoints, ShowLines : Boolean)
7435: Procedure ConRec( Canvas : TCanvas; Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
7436: Function Xpixel( X : Float) : Integer
7437: Function Ypixel( Y : Float) : Integer
7438: Function Xuser( X : Integer) : Float
7439: Function Yuser( Y : Integer) : Float
7440: end;
7441:
7442: Procedure FFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7443: Procedure IFFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7444: Procedure FFT_Integer( NumSamples : Integer; RealIn, ImagIn : TIntVector; OutArray : TCompVector)
7445: Procedure FFT_Integer_Cleanup
7446: Procedure CalcFrequency(NumSamples,FrequencyIndex: Integer;InArray: TCompVector;var FT : Complex)
7447: //unit uPSI_JclStreams;
7448: Function StreamSeek( Stream : TStream; const Offset : Int64; const Origin : TSeekOrigin) : Int64
7449: Function StreamCopy( Source : TStream; Dest : TStream; BufferSize : Integer) : Int64
7450: Function CompareStreams( A, B : TStream; BufferSize : Integer) : Boolean
7451: Function JCompareFiles( const FileA, FileB : TFileName; BufferSize : Integer) : Boolean
7452:
7453: procedure SIRegister_FmxUtils(CL: TPSPascalCompiler);
7454: begin
7455:   FindClass('TOBJECT'),'EInvalidDest
7456:   FindClass('TOBJECT'),'EFCantMove
7457:   Procedure fmxCopyFile( const FileName, DestName : string)
7458:   Procedure fmxMoveFile( const FileName, DestName : string)
7459:   Function fmxGetFileSize( const FileName : string) : LongInt
7460:   Function fmxFileDateTime( const FileName : string) : TDateTime
7461:   Function fmxHasAttr( const FileName : string; Attr : Word) : Boolean
7462:   Function fmxExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer):THandle;
7463: end;
7464:
7465: procedure SIRegister_FindFileIter(CL: TPSPascalCompiler);
7466: begin
7467:   SIRegister_IFindFileIterator(CL);
7468:   Function CreateFindFile(const Path:string; IncludeAttr:Integer;out iffi:IFindFileIterator):Bool;
7469: end;
7470:
7471: procedure SIRegister_PCharUtils(CL: TPSPascalCompiler);
7472: begin
7473:   Function SkipWhite( cp : PChar ) : PChar
7474:   Function ReadStringDoubleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7475:   Function ReadStringSingleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7476:   Function ReadIdent( cp : PChar; var ident : string) : PChar
7477: end;
7478:
7479: procedure SIRegister_JclStrHashMap(CL: TPSPascalCompiler);
7480: begin
7481:   SIRegister_TStringHashMapTraits(CL);
7482:   Function CaseSensitiveTraits : TStringHashMapTraits
7483:   Function CaseInsensitiveTraits : TStringHashMapTraits
7484:   THashNode', 'record Str : string; Ptr : Pointer; Left : PHashNod'
7485:   +'e; Right : PHashNode; end
7486:   //PHashArray', '^THashArray // will not work
7487:   SIRegister_TStringHashMap(CL);
7488:   THashValue', 'Cardinal
7489:   Function StrHash( const s : string) : THashValue
7490:   Function TextHash( const s : string) : THashValue
7491:   Function DataHash( var AValue, ASize : Cardinal) : THashValue
7492:   Function Iterate_FreeObjects( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7493:   Function Iterate_Dispose( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7494:   Function Iterate_FreeMem( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7495:   SIRegister_TCaseSensitiveTraits(CL);
7496:   SIRegister_TCaseInsensitiveTraits(CL);
7497:
7498:
7499: //*****unit uPSI_umath;
7500: Function uExp( X : Float) : Float
7501: Function uExp2( X : Float) : Float
7502: Function uExp10( X : Float) : Float

```

```

7503: Function uLog( X : Float ) : Float
7504: Function uLog2( X : Float ) : Float
7505: Function uLog10( X : Float ) : Float
7506: Function uLogA( X, A : Float ) : Float
7507: Function uIntPower( X : Float; N : Integer) : Float
7508: Function uPower( X, Y : Float ) : Float
7509: Function SgnGamma( X : Float ) : Integer
7510: Function Stirling( X : Float ) : Float
7511: Function StirLog( X : Float ) : Float
7512: Function Gamma( X : Float ) : Float
7513: Function LnGamma( X : Float ) : Float
7514: Function DiGamma( X : Float ) : Float
7515: Function TriGamma( X : Float ) : Float
7516: Function IGamma( X : Float ) : Float
7517: Function JGamma( X : Float ) : Float
7518: Function InvGamma( X : Float ) : Float
7519: Function Erf( X : Float ) : Float
7520: Function Erfc( X : Float ) : Float
7521: Function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7522: { Correlation coefficient between samples X and Y }
7523: function DBeta(A, B, X : Float) : Float;
7524: { Density of Beta distribution with parameters A and B }
7525: Function LambertW( X : Float; UBranch, Offset : Boolean) : Float
7526: Function Beta(X, Y : Float) : Float
7527: Function Binomial( N, K : Integer) : Float
7528: Function PBinom( N : Integer; P : Float; K : Integer) : Float
7529: Procedure Cholesky( A, L : TMatrix; Lb, Ub : Integer)
7530: Procedure LU_Decompo(A : TMatrix; Lb, Ub : Integer)
7531: Procedure LU_Solve( A : TMatrix; B : TVector; Lb, Ub : Integer; X : TVector)
7532: Function DNorm( X : Float ) : Float
7533:
7534: function DGamma(A, B, X : Float) : Float;
7535: { Density of Gamma distribution with parameters A and B }
7536: function DKhi2(Nu : Integer; X : Float) : Float;
7537: { Density of Khi-2 distribution with Nu d.o.f. }
7538: function DStudent(Nu : Integer; X : Float) : Float;
7539: { Density of Student distribution with Nu d.o.f. }
7540: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float;
7541: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
7542: function IBeta(A, B, X : Float) : Float;
7543: { Incomplete Beta function}
7544: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7545:
7546: procedure SIRegister_unlfit(CL: TPSPascalCompiler);
7547: begin
7548: Procedure SetOptAlgo( Algo : TOptAlgo)
7549: procedure SetOptAlgo(Algo : TOptAlgo);
7550: { -----
7551: Sets the optimization algorithm according to Algo, which must be
7552: NL_MARQ, NL_SIMP, NL_BFGS, NL_SA, NL_GA. Default is NL_MARQ
7553:
7554: Function GetOptAlgo : TOptAlgo
7555: Procedure SetMaxParam( N : Byte)
7556: Function GetMaxParam : Byte
7557: Procedure SetParamBounds( I : Byte; ParamMin, ParamMax : Float)
7558: Procedure GetParamBounds( I : Byte; var ParamMin, ParamMax : Float)
7559: Function NullParam( B : TVector; Lb, Ub : Integer) : Boolean
7560: Procedure NLFit( RegFunc : TRegFunc; DerivProc : TDervProc; X, Y : TVector; Lb, Ub : Integer; MaxIter : Integer; Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7561: Procedure WNLFit( RegFunc : TRegFunc; DerivProc : TDervProc; X, Y, S : TVector; Lb, Ub:Integer; MaxIter:Integer;Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7562: Procedure SetMCFile( FileName : String)
7563: Procedure SimFit(RegFunc:TRegFunc;X,Y:TVector;Lb,Ub:Integer;B:TVector;FirstPar,LastPar:Integer;V:TMatrix;
7564: Procedure WSimFit(RegFunc:TRegFunc; X,Y,S:TVector;Lb,Ub:Integer;B:TVector;FirstPar,
LastPar:Integer;V:TMatrix);
7565: end;
7566:
7567: (*-----*)
7568: procedure SIRegister_usimplex(CL: TPSPascalCompiler);
7569: begin
7570: Procedure SaveSimplex( FileName : string)
7571: Procedure Simplex(Func:TFuncNVar; X:TVector;Lb,Ub:Integer; MaxIter:Integer;Tol:Float; var F_min:Float);
7572: end;
7573: (*-----*)
7574: procedure SIRegister_uregtest(CL: TPSPascalCompiler);
7575: begin
7576: Procedure RegTest(Y,Ycalc: TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest)
7577: Procedure WRegTest(Y,Ycalc,S:TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest);
7578: end;
7579:
7580: procedure SIRegister_ustrings(CL: TPSPascalCompiler);
7581: begin
7582: Function LTrim( S : String ) : String
7583: Function RTrim( S : String ) : String
7584: Function uTrim( S : String ) : String
7585: Function StrChar( N : Byte; C : Char ) : String
7586: Function RFill( S : String; L : Byte) : String
7587: Function LFill( S : String; L : Byte) : String
7588: Function CFill( S : String; L : Byte) : String

```

```

7589: Function Replace( S : String; C1, C2 : Char ) : String
7590: Function Extract( S : String; var Index : Byte; Delim : Char ) : String
7591: Procedure Parse( S : String; Delim : Char; Field : TStrVector; var N : Byte )
7592: Procedure SetFormat( NumLength, MaxDec : Integer; FloatPoint, NSZero : Boolean )
7593: Function FloatToStr( X : Float ) : String
7594: Function IntToStr( N : LongInt ) : String
7595: Function uCompStr( Z : Complex ) : String
7596: end;
7597:
7598: procedure SIRegister_uhyper(CL: TPSPascalCompiler);
7599: begin
7600:   Function uSinh( X : Float ) : Float
7601:   Function uCosh( X : Float ) : Float
7602:   Function uTanh( X : Float ) : Float
7603:   Function uArcSinh( X : Float ) : Float
7604:   Function uArcCosh( X : Float ) : Float
7605:   Function ArcTanh( X : Float ) : Float
7606:   Procedure SinhCosh( X : Float; var SinhX, CoshX : Float )
7607: end;
7608:
7609: procedure SIRegister_urandom(CL: TPSPascalCompiler);
7610: begin
7611: type RNG_Type =
7612:   (RNG_MWC,           { Multiply-With-Carry }
7613:    RNG_MT,            { Mersenne Twister }
7614:    RNG_UVAG);        { Universal Virtual Array Generator }
7615: Procedure SetRNG( RNG : RNG_Type )
7616: Procedure InitGen( Seed : RNG_IntType )
7617: Procedure SRand( Seed : RNG_IntType )
7618: Function IRanGen : RNG_IntType
7619: Function IRanGen31 : RNG_IntType
7620: Function RanGen1 : Float
7621: Function RanGen2 : Float
7622: Function RanGen3 : Float
7623: Function RanGen53 : Float
7624: end;
7625:
7626: // Optimization by Simulated Annealing
7627: procedure SIRegister_usimann(CL: TPSPascalCompiler);
7628: begin
7629:   Procedure InitSAParams( NT, NS, NCycles : Integer; RT : Float )
7630:   Procedure SA_CreateLogFile( FileName : String )
7631:   Procedure SimAnn(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float );
7632: end;
7633:
7634: procedure SIRegister_uranuvag(CL: TPSPascalCompiler);
7635: begin
7636:   Procedure InitUVAGbyString( KeyPhrase : string )
7637:   Procedure InitUVAG( Seed : RNG_IntType )
7638:   Function IRanUVAG : RNG_IntType
7639: end;
7640:
7641: procedure SIRegister_ugenalg(CL: TPSPascalCompiler);
7642: begin
7643:   Procedure InitGAParams( NP, NG : Integer; SR, MR, HR : Float )
7644:   Procedure GA_CreateLogFile( LogFileName : String )
7645:   Procedure GenAlg(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float );
7646: end;
7647:
7648: TVector', 'array of Float
7649: procedure SIRegister_uqsort(CL: TPSPascalCompiler);
7650: begin
7651:   Procedure QSort( X : TVector; Lb, Ub : Integer )
7652:   Procedure DQSort( X : TVector; Lb, Ub : Integer )
7653: end;
7654:
7655: procedure SIRegister_uinterv(CL: TPSPascalCompiler);
7656: begin
7657:   Procedure Interval( X1, X2 : Float; MinDiv, MaxDiv : Integer; var Min, Max, Step : Float )
7658:   Procedure AutoScale(X: TVector; Lb, Ub : Integer; Scale : TScale; var XMin, XMax, XStep:Float )
7659: end;
7660:
7661: procedure SIRegister_D2XXUnit(CL: TPSPascalCompiler);
7662: begin
7663:   FT_Result', 'Integer
7664:   //TDWordptr', '^DWord // will not work
7665:   TFT_Program_Data', 'record Signature1 : DWord; Signature2 : DWor'
7666:   d; Version : DWord; VendorID : Word; ProductID : Word; Manufacturer : PChar'
7667:   r; ManufacturerID : PChar; Description : PChar; SerialNumber : PChar; MaxP'
7668:   ower : Word; PnP : Word; SelfPowered : Word; RemoteWakeUp : Word; Rev4 : B'
7669:   yte; IsoIn : Byte; IsoOut : Byte; PullDownEnable : Byte; SerNumEnable : By'
7670:   te; USBVersionEnable : Byte; USBVersion : Word; Rev5 : Byte; IsoInA : Byte'
7671:   ; IsoInB : Byte; IsoOutA : Byte; IsoOutB : Byte; PullDownEnable5 : Byte; S'
7672:   erNumEnable5 : Byte; USBVersionEnable5 : Byte; USBVersion5 : Word; AIsHigh'
7673:   Current : Byte; BIsHighCurrent : Byte; IFAIsFifo : Byte; IFAIsFifoTar : By'
7674:   te; IFAIsFastSer : Byte; AIsVCP : Byte; IFBIsFifo : Byte; IFBIsFifoTar : B'
7675:   yte; IFBIsFastSer : Byte; BIsVCP : Byte; UseExtOsc : Byte; HighDriveIOs : '
7676:   Byte; EndpointSize : Byte; PullDownEnableR : Byte; SerNumEnableR : Byte; I'
7677:   nvertTXD : Byte; InvertRXD : Byte; InvertRTS : Byte; InvertCTS : Byte; Inv'

```

```

7678:   ertDTR : Byte; InvertDSR : Byte; InvertDCD : Byte; InvertRI : Byte; Cbus0 :  

7679:   : Byte; Cbus1 : Byte; Cbus2 : Byte; Cbus3 : Byte; Cbus4 : Byte; RIsVCP : B'  

7680:   yte; end  

7681: end;  

7682:  

7683:  

7684: //***** PaintFX*****  

7685: procedure SIRegister_TJvPaintFX(CL: TPSPascalCompiler);  

7686: begin  

7687:   //with RegClassS(CL,'TComponent', 'TJvPaintFX') do  

7688:   with AddClassN(FindClass('TComponent'), 'TJvPaintFX') do begin  

7689:     Procedure Solarize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)  

7690:     Procedure Posterize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)  

7691:     Procedure Blend( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)  

7692:     Procedure Blend2( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)  

7693:     Procedure ExtractColor( const Dst : TBitmap; AColor : TColor)  

7694:     Procedure ExcludeColor( const Dst : TBitmap; AColor : TColor)  

7695:     Procedure Turn( Src, Dst : TBitmap)  

7696:     Procedure TurnRight( Src, Dst : TBitmap)  

7697:     Procedure HeightMap( const Dst : TBitmap; Amount : Integer)  

7698:     Procedure TexturizeTile( const Dst : TBitmap; Amount : Integer)  

7699:     Procedure TexturizeOverlap( const Dst : TBitmap; Amount : Integer)  

7700:     Procedure RippleRandom( const Dst : TBitmap; Amount : Integer)  

7701:     Procedure RippleTooth( const Dst : TBitmap; Amount : Integer)  

7702:     Procedure RippleTriangle( const Dst : TBitmap; Amount : Integer)  

7703:     Procedure Triangles( const Dst : TBitmap; Amount : Integer)  

7704:     Procedure DrawMandelJulia(const Dst: TBitmap; x0,y0,x1,y1:Single; Niter:Integer; Mandel:Boolean)  

7705:     Procedure FilterXBlue( const Dst : TBitmap; Min, Max : Integer)  

7706:     Procedure FilterXGreen( const Dst : TBitmap; Min, Max : Integer)  

7707:     Procedure FilterXRed( const Dst : TBitmap; Min, Max : Integer)  

7708:     Procedure FilterBlue( const Dst : TBitmap; Min, Max : Integer)  

7709:     Procedure FilterGreen( const Dst : TBitmap; Min, Max : Integer)  

7710:     Procedure FilterRed( const Dst : TBitmap; Min, Max : Integer)  

7711:     Procedure Emboss( var Bmp : TBitmap)  

7712:     Procedure Plasma( Src1, Src2, Dst : TBitmap; Scale, Turbulence : Single)  

7713:     Procedure Shake( Src, Dst : TBitmap; Factor : Single)  

7714:     Procedure ShakeDown( Src, Dst : TBitmap; Factor : Single)  

7715:     Procedure KeepBlue( const Dst : TBitmap; Factor : Single)  

7716:     Procedure KeepGreen( const Dst : TBitmap; Factor : Single)  

7717:     Procedure KeepRed( const Dst : TBitmap; Factor : Single)  

7718:     Procedure Mandelbrot( const Dst : TBitmap; Factor : Integer)  

7719:     Procedure MaskMandelbrot( const Dst : TBitmap; Factor : Integer)  

7720:     Procedure FoldRight( Src1, Src2, Dst : TBitmap; Amount : Single)  

7721:     Procedure QuartoOpaque( Src, Dst : TBitmap)  

7722:     Procedure SemiOpaque( Src, Dst : TBitmap)  

7723:     Procedure ShadowDownLeft( const Dst : TBitmap)  

7724:     Procedure ShadowDownRight( const Dst : TBitmap)  

7725:     Procedure ShadowUpLeft( const Dst : TBitmap)  

7726:     Procedure ShadowUpRight( const Dst : TBitmap)  

7727:     Procedure Darkness( const Dst : TBitmap; Amount : Integer)  

7728:     Procedure Trace( const Dst : TBitmap; Intensity : Integer)  

7729:     Procedure FlipRight( const Dst : TBitmap)  

7730:     Procedure FlipDown( const Dst : TBitmap)  

7731:     Procedure Spotlight( const Dst : TBitmap; Amount : Integer; Spot : TRect)  

7732:     Procedure SplitLight( const Dst : TBitmap; Amount : Integer)  

7733:     Procedure MakeSeamlessClip( var Dst : TBitmap; Seam : Integer)  

7734:     Procedure Wave( const Dst : TBitmap; Amount, Inference, Style : Integer)  

7735:     Procedure Mosaic( const Bm : TBitmap; Size : Integer)  

7736:     Procedure SmoothRotate( var Src, Dst : TBitmap; CX, CY : Integer; Angle : Single)  

7737:     Procedure SmoothResize( var Src, Dst : TBitmap)  

7738:     Procedure Twist( var Bmp, Dst : TBitmap; Amount : Integer)  

7739:     Procedure SplitBlur( const Dst : TBitmap; Amount : Integer)  

7740:     Procedure GaussianBlur( const Dst : TBitmap; Amount : Integer)  

7741:     Procedure Smooth( const Dst : TBitmap; Weight : Integer)  

7742:     Procedure GrayScale( const Dst : TBitmap)  

7743:     Procedure AddColorNoise( const Dst : TBitmap; Amount : Integer)  

7744:     Procedure AddMonoNoise( const Dst : TBitmap; Amount : Integer)  

7745:     Procedure Contrast( const Dst : TBitmap; Amount : Integer)  

7746:     Procedure Lightness( const Dst : TBitmap; Amount : Integer)  

7747:     Procedure Saturation( const Dst : TBitmap; Amount : Integer)  

7748:     Procedure Spray( const Dst : TBitmap; Amount : Integer)  

7749:     Procedure AntiAlias( const Dst : TBitmap)  

7750:     Procedure AntiAliasRect( const Dst : TBitmap; XOrigin, YOrigin, XFinal, YFinal : Integer)  

7751:     Procedure SmoothPoint( const Dst : TBitmap; XK, YK : Integer)  

7752:     Procedure FishEye( var Bmp, Dst : TBitmap; Amount : Single)  

7753:     Procedure Marble( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)  

7754:     Procedure Marble2( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)  

7755:     Procedure Marble3( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)  

7756:     Procedure Marble4( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)  

7757:     Procedure Marble5( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)  

7758:     Procedure Marble6( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)  

7759:     Procedure Marble7( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)  

7760:     Procedure Marble8( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)  

7761:     Procedure SqueezeHor( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)  

7762:     Procedure SplitRound( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)  

7763:     Procedure Tile( Src, Dst : TBitmap; Amount : Integer)  

7764:     Procedure Stretch( Src, Dst : TBitmap; Filter : TFilterProc; AWidth : Single)  

7765:     Procedure Grow( Src1, Src2, Dst : TBitmap; Amount : Single; X, Y : Integer)  

7766:     Procedure Invert( Src : TBitmap)

```

```

7767:     Procedure MirrorRight( Src : TBitmap )
7768:     Procedure MirrorDown( Src : TBitmap )
7769:   end;
7770: end;
7771:
7772: (*-----*)
7773: procedure SIRegister_JvPaintFX(CL: TPSPascalCompiler);
7774: begin
7775:   AddTypeS('TLightBrush', '( lbBrightness, lbContrast, lbSaturation, lbFishe'
7776:             +'ye, lrotate, lbtwist, lbrimble, mbHor, mbTop, mbBottom, mbDiamond, mbWast'
7777:             +'e, mbRound, mbRound2, mbSplitRound, mbSplitWaste )'
7778:             +SIRegister_TJvPaintFX(CL);
7779:   Function SplineFilter( Value : Single ) : Single
7780:   Function BellFilter( Value : Single ) : Single
7781:   Function TriangleFilter( Value : Single ) : Single
7782:   Function BoxFilter( Value : Single ) : Single
7783:   Function HermiteFilter( Value : Single ) : Single
7784:   Function Lanczos3Filter( Value : Single ) : Single
7785:   Function MitchellFilter( Value : Single ) : Single
7786: end;
7787:
7788:
7789: (*-----*)
7790: procedure SIRegister_Chart(CL: TPSPascalCompiler);
7791: begin
7792:   'TeeMsg_DefaultFunctionName', 'String 'TeeFunction
7793:   TeeMsg_DefaultSeriesName', 'String 'Series
7794:   TeeMsg_DefaultToolName', 'String 'ChartTool
7795:   ChartComponentPalette', 'string 'TeeChart
7796:   TeeMaxLegendColumns', LongInt'( 2 );
7797:   TeeDefaultLegendSymbolWidth', 'LongInt'( 20 );
7798:   TeeTitleFootDistance, LongInt( 5 );
7799:   SIRegister_TCustomChartWall(CL);
7800:   SIRegister_TChartWall(CL);
7801:   SIRegister_TChartLegendGradient(CL);
7802:   TLegendStyle', '( lsAuto, lsSeries, lsValues, lsLastValues, lsSeriesGroups )'
7803:   TLegendAlignment', '( laLeft, laRight, laTop, laBottom )'
7804:   FindClass('TOBJECT'), 'LegendException
7805:   TOnGetLegendText', 'Procedure ( Sender : TCustomAxisPanel; Legen'
7806:             +'dStyle : TLegendStyle; Index : Integer; var LegendText : String )
7807:   FindClass('TOBJECT'), 'TCustomChartLegend
7808:   TLegendSymbolSize', '( lcsPercent, lcsPixels )'
7809:   TLegendSymbolPosition', '( spLeft, spright )'
7810:   TSymbolDrawEvent', 'Procedure(Sender:TObject;Series:TChartSeries;ValueIndex:Integer;R:TRect)'
7811:   TSymbolCalcHeight', 'Function : Integer
7812:   SIRegister_TLegendSymbol(CL);
7813:   SIRegister_TTeeCustomShapePosition(CL);
7814:   TCheckBoxesStyle', '( cbsCheck, cbsRadio )'
7815:   SIRegister_TLegendTitle(CL);
7816:   SIRegister_TLegendItem(CL);
7817:   SIRegister_TLegendItems(CL);
7818:   TLegendCalcSize', 'Procedure ( Sender : TCustomChartLegend; var ASize : Integer )
7819:   FindClass('TOBJECT'), 'TCustomChart
7820:   SIRegister_TCustomChartLegend(CL);
7821:   SIRegister_TChartLegend(CL);
7822:   SIRegister_TChartTitle(CL);
7823:   SIRegister_TChartFootTitle(CL);
7824:   TChartClick', 'Procedure ( Sender : TCustomChart; Button : TMous'
7825:             +'eButton; Shift : TShiftState; X, Y : Integer )
7826:   TChartClickAxis', 'Procedure ( Sender : TCustomChart; Axis : TCh'
7827:             +'artAxis; Button : TMouseButton; Shift : TShiftState; X, Y : Integer )
7828:   TChartClickSeries', 'Procedure ( Sender : TCustomChart; Series : '
7829:             +TChartSeries; ValueIndex : Integer; Button: TMouseButton; Shift:TShiftState;X,Y:Integer )
7830:   TChartClickTitle', 'Procedure ( Sender : TCustomChart; ATitle : '
7831:             +'TChartTitle; Button : TMouseButton; Shift : TShiftState; X, Y : Integer )
7832:   TOnGetLegendPos', 'Procedure ( Sender: TCustomChart; Index: Integer; var X,Y,XColor:Integer )
7833:   TOnGetLegendRect', 'Procedure ( Sender : TCustomChart; var Rect : TRect )
7834:   TAxissavedScales', 'record Auto : Boolean; AutoMin : Boolean; Au'
7835:             +'toMax : Boolean; Min : Double; Max : Double; end
7836:   TAxissavedScales', 'array of TAxissavedScales
7837:   SIRegister_TChartBackWall(CL);
7838:   SIRegister_TChartRightWall(CL);
7839:   SIRegister_TChartBottomWall(CL);
7840:   SIRegister_TChartLeftWall(CL);
7841:   SIRegister_TChartWalls(CL);
7842:   TChartAllowScrollEvent', 'Procedure(Sender:TChartAxis;var AMin,AMax:Double;var AllowScroll:Boolean);'
7843:   SIRegister_TCustomChart(CL);
7844:   SIRegister_TChart(CL);
7845:   SIRegister_TTeeSeriesTypes(CL);
7846:   SIRegister_TTeeToolTypes(CL);
7847:   SIRegister_TTeeDragObject(CL);
7848:   SIRegister_TColorPalettes(CL);
7849:   Procedure RegisterTeeSeries(ASeriesClass:TChartSeriesClass;ADescription,
    AGalleryPage:PString;ANumGallerySeries:Integer;
7850:   Procedure RegisterTeeSeries1( ASeriesClass : TChartSeriesClass; ADscription : PString );
7851:   Procedure RegisterTeeFunction(AFunctionClass:T TeeFunctionClass;ADescription,
    AGalleryPage:PString;ANumGallerySeries : Int;
7852:   Procedure RegisterTeeBasicFunction( AFunctionClass : TTeeFunctionClass; ADscription : PString )
7853:   Procedure RegisterTeeSeriesFunction(ASeriesClass: TChartSeriesClass;AFunctionClass:TTeeFunctionClass;
    ADscription, AGalleryPage : PString; ANumGallerySeries : Integer; ASubIndex : Integer)

```

```

7854: Procedure UnRegisterTeeSeries( const ASeriesList : array of TChartSeriesClass )
7855: Procedure UnRegisterTeeFunctions( const AFunctionList : array of TTeeFunctionClass )
7856: Procedure AssignSeries( var OldSeries, NewSeries : TChartSeries )
7857: Function CreateNewTeeFunction( ASeries : TChartSeries; AClass : TTeeFunctionClass ) : TTeeFunction
7858: Function CreateNewSeries( AOwner : TComponent; AChart : TCustomAxisPanel; AClass : TChartSeriesClass;
    AFunctionClass : TTeeFunctionClass ) : TChartSeries
7859: Function CloneChartSeries( ASeries : TChartSeries ) : TChartSeries;
7860: Function CloneChartSeries1( ASeries : TChartSeries; AChart : TCustomAxisPanel ) : TChartSeries;
7861: Function CloneChartSeries2( ASeries : TChartSeries; AOwner : TComponent; AChart : TCustomAxisPanel ) : TChartSeries;;
7862: Function CloneChartTool( ATool : TTeeCustomTool; AOwner : TComponent ) : TTeeCustomTool
7863: Function ChangeSeriesType( var ASeries : TChartSeries; NewType : TChartSeriesClass ) : TChartSeries
7864: Procedure ChangeAllSeriesType( AChart : TCustomChart; AClass : TChartSeriesClass )
7865: Function GetNewSeriesName( AOwner : TComponent ) : TComponentName
7866: Procedure RegisterTeeTools( const ATools : array of TTeeCustomToolClass )
7867: Procedure UnRegisterTeeTools( const ATools : array of TTeeCustomToolClass )
7868: Function GetGallerySeriesName( ASeries : TChartSeries ) : String
7869: Procedure PaintSeriesLegend( ASeries : TChartSeries; ACanvas : TCanvas; const
    R : TRect; ReferenceChart : TCustomChart );
7870: SIRegister_TChartTheme( CL );
7871: //TChartThemeClass', 'class of TChartTheme
7872: //TCanvasClass', 'class of TCanvas3D
7873: Function SeriesNameOrIndex( ASeries : TCustomChartSeries ) : String
7874: Function SeriesTitleOrName( ASeries : TCustomChartSeries ) : String
7875: Procedure FillSeriesItems( AItems : TStringList; AList : TCustomSeriesList; UseTitles : Boolean )
7876: Procedure ShowMessageUser( const S : String )
7877: Function HasNoMandatoryValues( ASeries : TChartSeries ) : Boolean
7878: Function HasLabels( ASeries : TChartSeries ) : Boolean
7879: Function HasColors( ASeries : TChartSeries ) : Boolean
7880: Function SeriesGuessContents( ASeries : TChartSeries ) : TeeFormatFlag
7881: Procedure TeeDrawBitmapEditor( Canvas : TCanvas; Element : TCustomChartElement; Left, Top : Integer )
7882: end;
7883:
7884:
7885: procedure SIRegister_TeeProcs( CL : TPPascalCompiler );
7886: begin
7887: // 'TeeFormBorderStyle', 'bsNone';
7888: SIRegister_TMetafile( CL );
7889: 'TeeDefVerticalMargin', 'LongInt'( 4 );
7890: 'TeeDefHorizMargin', 'LongInt'( 3 );
7891: 'crTeeHand', 'LongInt'( TCursor( 2020 ) );
7892: 'TeeMsg_TeeHand', 'String' 'crTeeHand'
7893: 'TeeNormalPrintDetail', 'LongInt'( 0 );
7894: 'TeeHighPrintDetail', 'LongInt'( - 100 );
7895: 'TeeDefault_PrintMargin', 'LongInt'( 15 );
7896: 'MaxDefaultColors', 'LongInt'( 19 );
7897: 'TeeTabDelimiter', 'Char #9';
7898: 'TDateTimeStep', '( dtOneMicroSecond, dtOneMillisecond, dtOneSeco'
7899: + 'nd, dtFiveSeconds, dtTenSeconds, dtFifteenSeconds, dtThirtySeconds, dtOneM'
7900: + 'inute, dtFiveMinutes, dtTenMinutes, dtFifteenMinutes, dtThirtyMinutes, dtO'
7901: + 'neHour, dtTwoHours, dtSixHours, dtTwelveHours, dtOneDay, dtTwoDays, dtThre'
7902: + 'eDays, dtOneWeek, dtHalfMonth, dtOneMonth, dtTwoMonths, dtThreeMonths, dtF'
7903: + 'ourMonths, dtSixMonths, dtOneYear, dtNone )'
7904: SIRegister_TCustomPanelNoCaption( CL );
7905: FindClass( 'TOBJECT' ), 'TCustomTeePanel'
7906: SIRegister_TZoomPanning( CL );
7907: SIRegister_TTeeEvent( CL );
7908: //SIRegister_TTeeEventListeners( CL );
7909: TTeeMouseEventKind', '( meDown, meUp, meMove )'
7910: SIRegister_TTeeMouseEvent( CL );
7911: SIRegister_TCustomTeePanel( CL );
7912: //TChartGradient', 'TTeeGradient
7913: //TChartGradientClass', 'class of TChartGradient
7914: TPanningMode', '( pmNone, pmHorizontal, pmVertical, pmBoth )'
7915: SIRegister_TTeeZoomPen( CL );
7916: SIRegister_TTeeZoomBrush( CL );
7917: TteeZoomDirection', '( tzdHorizontal, tzdVertical, tzdBoth )'
7918: SIRegister_TTeeZoom( CL );
7919: FindClass( 'TOBJECT' ), 'TCustomTeePanelExtended
7920: TTeeBackImageMode', '( pbmStretch, pbmTile, pbmCenter, pbmCustom )'
7921: SIRegister_TBackImage( CL );
7922: SIRegister_TCustomTeePanelExtended( CL );
7923: //TChartBrushClass', 'class of TChartBrush
7924: SIRegister_TTeeCustomShapeBrushPen( CL );
7925: TChartObjectShapeStyle', '( fosRectangle, fosRoundRectangle, fosEllipse )'
7926: TTextFormat', '( ttfNormal, ttfHtml )'
7927: SIRegister_TTeeCustomShape( CL );
7928: SIRegister_TTeeShape( CL );
7929: SIRegister_TTeeExportData( CL );
7930: Function TeeStr( const Num : Integer ) : String
7931: Function DateTimeDefaultFormat( const AStep : Double ) : String
7932: Function TEEDaysInMonth( Year, Month : Word ) : Word
7933: Function FindDateTimeStep( const StepValue : Double ) : TDateTimeStep
7934: Function NextDateTimeStep( const AStep : Double ) : Double
7935: Function PointInLine( const P : TPoint; const px, py, qx, qy : Integer ) : Boolean;
7936: Function PointInLine1( const P, FromPoint, ToPoint : TPoint ) : Boolean;
7937: Function PointInLine2( const P, FromPoint, ToPoint : TPoint; const TolerancePixels : Integer ) : Boolean;
7938: Function PointInLine3( const P : TPoint; const px, py, qx, qy, TolerancePixels : Integer ) : Boolean;
7939: Function PointInLineTolerance( const P : TPoint; const px, py, qx, qy, TolerancePixels : Integer ) : Boolean;
7940: Function PointInPolygon( const P : TPoint; const Poly : array of TPoint ) : Boolean

```

```

7941: Function PointInTriangle( const P, P0, P1, P2 : TPoint ) : Boolean;
7942: Function PointInTriangle1( const P : TPoint; X0, Y0, X1, Y1 : Integer ) : Boolean;
7943: Function PointInHorizTriangle( const P : TPoint; Y0, Y1, X0, X1 : Integer ) : Boolean;
7944: Function PointInEllipse( const P : TPoint; const Rect : TRect ) : Boolean;
7945: Function PointInEllipse1( const P : TPoint; Left, Top, Right, Bottom : Integer ) : Boolean;
7946: Function DelphiToLocalFormat( const Format : String ) : String;
7947: Function LocalToDelphiFormat( const Format : String ) : String;
7948: Procedure TEEEnableControls(Enable: Boolean; const ControlArray : array of TControl);
7949: Function TeeRoundDate(const ADate: TDateTime; AStep : TDateTimeStep) : TDateTime;
7950: Procedure TeeDateTimeIncrement(IsDateTime:Boolean;Increment:Double;var Value:Double;const
AnIncrement:Double; tmpWhichDateTime:TDateTimeStep)
7951: 'TeeSortCompare', 'Function ( a, b : Integer ) : Integer
7952: TTeeSortSwap', 'Procedure ( a, b : Integer )
7953: Procedure TeeSort(StartIndex,EndIndex:Integer;CompareFunc:TeeSortCompare;SwapFunc:TeeSortSwap);
7954: Function TeeGetUniqueName( AOwner : TComponent; const AStartName : String ) : string;
7955: Function TeeExtractField( St : String; Index : Integer ) : String;
7956: Function TeeExtractField1( St : String; Index : Integer; const Separator : String ) : String;
7957: Function TeeNumFields( St : String ) : Integer;
7958: Function TeeNumFields1( const St, Separator : String ) : Integer;
7959: Procedure TeeGetBitmapEditor( AObject : TObject; var Bitmap : TBitmap );
7960: Procedure TeeLoadBitmap( Bitmap : TBitmap; const Name1, Name2 : String );
7961: // TColorArray', 'array of TColor
7962: Function GetDefaultColor( const Index : Integer ) : TColor;
7963: Procedure SetDefaultColorPalette;
7964: Procedure SetDefaultColorPalette1( const Palette : array of TColor );
7965: 'TeeCheckBoxSize','LongInt'( 11 );
7966: Procedure TeeDrawCheckBox(x,y:Integer;Canvas:TCanvas;Checked:Boolean;ABackColor:TColor;CheckBox:Boolean);
7967: Function TEEStrToFloatDef( const S : string; const Default : Extended ) : Extended;
7968: Function TryStrToFloat( const S : string; var Value : Double ) : Boolean;
7969: Function CrossingLines( const X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Double; out x, y : Double ) : Boolean;
7970: Procedure TeeTranslateControl( AControl : TControl );
7971: Procedure TeeTranslateControl1( AControl : TControl; const ExcludeChilds : array of TControl );
7972: Function ReplaceChar( const AString : String; const Search : Char; const Replace : Char ) : String;
7973: //Procedure RectToFourPoints( const ARect : TRect; const Angle : Double; var P : TFourPoints );
7974: Function TeeAntiAlias( Panel : TCustomTeePanel; ChartRect : Boolean ) : TBitmap;
7975: //Procedure DrawBevel(Canvas:TTeeCanvas;Bevel:TPanelBevel;var R:TRect;Width:Integer;Round:Integer);
7976: //Function ScreenRatio( ACanvas : TCanvas3D ) : Double;
7977: Function TeeReadBoolOption( const AKey : String; DefaultValue : Boolean ) : Boolean;
7978: Procedure TeeSaveBoolOption( const AKey : String; Value : Boolean );
7979: Function TeeReadIntegerOption( const AKey : String; DefaultValue : Integer ) : Integer;
7980: Procedure TeeSaveIntegerOption( const AKey : String; Value : Integer );
7981: Function TeeReadStringOption( const AKey, DefaultValue : String ) : String;
7982: Procedure TeeSaveStringOption( const AKey, Value : String );
7983: Function TeeDefaultXMLEncoding : String;
7984: Procedure ConvertTextToXML( Stream : TStream; XMLHeader : Boolean );
7985: TeeWindowHandle', 'Integer;
7986: Procedure TeeGoToURL( Handle : TeeWindowHandle; const URL : String );
7987: Procedure HtmlTextOut( ACanvas : TCanvas; x, y : Integer; Text : String );
7988: Function HtmlTextExtent( ACanvas : TCanvas; const Text : String ) : TSize;
7989: end;
7990;
7991;
7992: using mXBDEUtils
7993: ****
7994: Procedure SetAlias( aAlias, aDirectory : String );
7995: Procedure CheckRegistryEntry(Reg:TRegistry;const Path,Value:String;const Default,
Desired:Variant;Size:Byte);
7996: Function GetFileVersionNumber( const FileName : String ) : TVersionNo;
7997: Procedure SetBDE( aPath, aNode, aValue : String );
7998: function RestartDialog(Wnd: HWnd; Reason: PChar; Flags: Integer): Integer; stdcall;
7999: Function GetSystemDirectory( lpBuffer : string; uSize : UINT ) : UINT;
8000: Function GetSystemDirectoryW( lpBuffer : pchar; uSize : UINT ) : UINT;
8001: Function GetTempPath( nBufferLength : DWORD; lpBuffer : string ) : DWORD;
8002: Function GetWindowsDirectoryW( nBufferLength : DWORD; lpBuffer : string ) : DWORD;
8003: Function GetTempFileName(lpPathName,lpPrefixString:string;uUnique:UINT;lpTempFileName:string):UINT;
8004;
8005;
8006: procedure SIRegister_cDateTime(CL: TPPSPascalCompiler);
8007: begin
8008: AddClassN(FindClass('TOBJECT'),'EDateTime';
8009: Function DatePart( const D : TDateTime ) : Integer;
8010: Function TimePart( const D : TDateTime ) : Double;
8011: Function Century( const D : TDateTime ) : Word;
8012: Function Year( const D : TDateTime ) : Word;
8013: Function Month( const D : TDateTime ) : Word;
8014: Function Day( const D : TDateTime ) : Word;
8015: Function Hour( const D : TDateTime ) : Word;
8016: Function Minute( const D : TDateTime ) : Word;
8017: Function Second( const D : TDateTime ) : Word;
8018: Function Millisecond( const D : TDateTime ) : Word;
8019: ('OneDay','Extended').SetExtended( 1.0 );
8020: ('OneHour','Extended').SetExtended( OneDay / 24 );
8021: ('OneMinute','Extended').SetExtended( OneHour / 60 );
8022: ('OneSecond','Extended').SetExtended( OneMinute / 60 );
8023: ('OneMillisecond','Extended').SetExtended( OneSecond / 1000 );
8024: ('OneWeek','Extended').SetExtended( OneDay * 7 );
8025: ('HoursPerDay','Extended').SetExtended( 24 );
8026: ('MinutesPerHour','Extended').SetExtended( 60 );
8027: ('SecondsPerMinute','Extended').SetExtended( 60 );

```

```

8028: Procedure SetYear( var D : TDateTime; const Year : Word)
8029: Procedure SetMonth( var D : TDateTime; const Month : Word)
8030: Procedure SetDay( var D : TDateTime; const Day : Word)
8031: Procedure SetHour( var D : TDateTime; const Hour : Word)
8032: Procedure SetMinute( var D : TDateTime; const Minute : Word)
8033: Procedure SetSecond( var D : TDateTime; const Second : Word)
8034: Procedure SetMillisecond( var D : TDateTime; const Milliseconds : Word)
8035: Function IsEqual( const D1, D2 : TDateTime) : Boolean;
8036: Function IsEqual1( const D1 : TDateTime; const Ye, Mo, Da : Word):Boolean;
8037: Function IsEqual2( const D1 : TDateTime; const Ho, Mi, Se, ms : Word):Boolean;
8038: Function IsAM( const D : TDateTime) : Boolean
8039: Function IsPM( const D : TDateTime) : Boolean
8040: Function IsMidnight( const D : TDateTime) : Boolean
8041: Function IsNoon( const D : TDateTime) : Boolean
8042: Function IsSunday( const D : TDateTime) : Boolean
8043: Function IsMonday( const D : TDateTime) : Boolean
8044: Function IsTuesday( const D : TDateTime) : Boolean
8045: Function IsWednesday( const D : TDateTime) : Boolean
8046: Function IsThursday( const D : TDateTime) : Boolean
8047: Function IsFriday( const D : TDateTime) : Boolean
8048: Function IsSaturday( const D : TDateTime) : Boolean
8049: Function IsWeekend( const D : TDateTime) : Boolean
8050: Function Noon( const D : TDateTime) : TDateTime
8051: Function Midnight( const D : TDateTime) : TDateTime
8052: Function FirstDayOfMonth( const D : TDateTime) : TDateTime
8053: Function LastDayOfMonth( const D : TDateTime) : TDateTime
8054: Function NextWorkday( const D : TDateTime) : TDateTime
8055: Function PreviousWorkday( const D : TDateTime) : TDateTime
8056: Function FirstDayOfYear( const D : TDateTime) : TDateTime
8057: Function LastDayOfYear( const D : TDateTime) : TDateTime
8058: Function EasterSunday( const Year : Word) : TDateTime
8059: Function GoodFriday( const Year : Word) : TDateTime
8060: Function AddMilliseconds( const D : TDateTime; const N : Int64) : TDateTime
8061: Function AddSeconds( const D : TDateTime; const N : Int64) : TDateTime
8062: Function AddMinutes( const D : TDateTime; const N : Integer) : TDateTime
8063: Function AddHours( const D : TDateTime; const N : Integer) : TDateTime
8064: Function AddDays( const D : TDateTime; const N : Integer) : TDateTime
8065: Function AddWeeks( const D : TDateTime; const N : Integer) : TDateTime
8066: Function AddMonths( const D : TDateTime; const N : Integer) : TDateTime
8067: Function AddYears( const D : TDateTime; const N : Integer) : TDateTime
8068: Function DayOfYear( const Ye, Mo, Da : Word) : Integer
8069: Function DayOfYear( const D : TDateTime) : Integer
8070: Function DaysInMonth( const Ye, Mo : Word) : Integer
8071: Function DaysInMonth( const D : TDateTime) : Integer
8072: Function DaysInYear( const Ye : Word) : Integer
8073: Function DaysInYearDate( const D : TDateTime) : Integer
8074: Function WeekNumber( const D : TDateTime) : Integer
8075: Function ISOFirstWeekOfYear( const Ye : Word) : TDateTime
8076: Procedure ISOWeekNumber( const D : TDateTime; var WeekNumber, WeekYear : Word)
8077: Function DiffMilliseconds( const D1, D2 : TDateTime) : Int64
8078: Function DiffSeconds( const D1, D2 : TDateTime) : Integer
8079: Function DiffMinutes( const D1, D2 : TDateTime) : Integer
8080: Function DiffHours( const D1, D2 : TDateTime) : Integer
8081: Function DiffDays( const D1, D2 : TDateTime) : Integer
8082: Function DiffWeeks( const D1, D2 : TDateTime) : Integer
8083: Function DiffMonths( const D1, D2 : TDateTime) : Integer
8084: Function DiffYears( const D1, D2 : TDateTime) : Integer
8085: Function GMTBias : Integer
8086: Function GMTimeToLocalTime( const D : TDateTime) : TDateTime
8087: Function LocalTimeToGMTime( const D : TDateTime) : TDateTime
8088: Function NowAsGMTime : TDateTime
8089: Function DateimeToISO8601String( const D : TDateTime) : AnsiString
8090: Function ISO8601StringToTime( const D : AnsiString) : TDateTime
8091: Function ISO8601StringAsDateTime( const D : AnsiString) : TDateTime
8092: Function DateTimeToANSI( const D : TDateTime) : Integer
8093: Function ANSIToDateime( const Julian : Integer) : TDateTime
8094: Function DateTimeToISOInteger( const D : TDateTime) : Integer
8095: Function DateTimeToISOString( const D : TDateTime) : AnsiString
8096: Function ISOIntegerToDateime( const ISOInteger : Integer) : TDateTime
8097: Function TwoDigitRadix2000YearToYear( const Y : Integer) : Integer
8098: Function DatetimeAsElapsedTime(const D:TDateTime; const IncludeMilliseconds:Boolean):AnsiString
8099: Function UnixTimeToDateime( const UnixTime : LongWord) : TDateTime
8100: Function DatetimeToUnixTime( const D : TDateTime) : LongWord
8101: Function EnglishShortDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8102: Function EnglishShortDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8103: Function EnglishLongDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8104: Function EnglishLongDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8105: Function EnglishShortMonthStrA( const Month : Integer) : AnsiString
8106: Function EnglishShortMonthStrU( const Month : Integer) : UnicodeString
8107: Function EnglishLongMonthStrA( const Month : Integer) : AnsiString
8108: Function EnglishLongMonthStrU( const Month : Integer) : UnicodeString
8109: Function EnglishShortDayOfWeekA( const S : AnsiString) : Integer
8110: Function EnglishShortDayOfWeekU( const S : UnicodeString) : Integer
8111: Function EnglishLongDayOfWeekA( const S : AnsiString) : Integer
8112: Function EnglishLongDayOfWeekU( const S : UnicodeString) : Integer
8113: Function EnglishShortMonthA( const S : AnsiString) : Integer
8114: Function EnglishShortMonthU( const S : UnicodeString) : Integer
8115: Function EnglishLongMonthA( const S : AnsiString) : Integer
8116: Function EnglishLongMonthU( const S : UnicodeString) : Integer

```

```

8117: Function RFC850DayOfWeekA( const S : AnsiString ) : Integer
8118: Function RFC850DayOfWeekU( const S : UnicodeString ) : Integer
8119: Function RFC1123DayOfWeekA( const S : AnsiString ) : Integer
8120: Function RFC1123DayOfWeekU( const S : UnicodeString ) : Integer
8121: Function RFCMonthA( const S : AnsiString ) : Word
8122: Function RFCMonthU( const S : UnicodeString ) : Word
8123: Function GMTTimeToRFC1123TimeA( const D : TDateTime; const IncludeSeconds:Boolean ) : AnsiString
8124: Function GMTTimeToRFC1123TimeU( const D : TDateTime; const IncludeSeconds:Boolean ) : UnicodeString
8125: Function GMTDateTimeToRFC1123DateTimeA( const D: TDateTime; const IncludeDayOfWeek:Bool ):AnsiString;
8126: Function GMTDateTimeToRFC1123DateTimeU( const D:TDateTime;const IncludeDayOfWeek:Bool ):UnicodeString;
8127: Function DateTimeToRFCDateTimeA( const D : TDateTime ) : AnsiString
8128: Function DateTimeToRFCDateTimeU( const D : TDateTime ) : UnicodeString
8129: Function NowAsRFCDateTimeA : AnsiString
8130: Function NowAsRFCDateTimeU : UnicodeString
8131: Function RFCDateTimeToGMTDateTime( const S : AnsiString ) : TDateTime
8132: Function RFCDateTimeToDateTIme( const S : AnsiString ) : TDateTime
8133: Function RFCTimeZoneToGMTBias( const Zone : AnsiString ) : Integer
8134: Function TimePeriodStr( const D : TDateTime ) : AnsiString
8135: Procedure SelfTest
8136: end;
8137: //*****CFileUtils
8138: Function PathHasDriveLetterA( const Path : AnsiString ) : Boolean
8139: Function PathHasDriveLetter( const Path : String ) : Boolean
8140: Function PathIsDriveLetterA( const Path : AnsiString ) : Boolean
8141: Function PathIsDriveLetter( const Path : String ) : Boolean
8142: Function PathIsDriveRootA( const Path : AnsiString ) : Boolean
8143: Function PathIsDriveRoot( const Path : String ) : Boolean
8144: Function PathIsRootA( const Path : AnsiString ) : Boolean
8145: Function PathIsRoot( const Path : String ) : Boolean
8146: Function PathIsUNCPathA( const Path : AnsiString ) : Boolean
8147: Function PathIsUNCPath( const Path : String ) : Boolean
8148: Function PathIsAbsoluteA( const Path : AnsiString ) : Boolean
8149: Function PathIsAbsolute( const Path : String ) : Boolean
8150: Function PathIsDirectoryA( const Path : AnsiString ) : Boolean
8151: Function PathIsDirectory( const Path : String ) : Boolean
8152: Function PathInclSuffixA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8153: Function PathInclSuffix( const Path : String; const PathSep : Char ) : String
8154: Function PathExclSuffixA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8155: Function PathExclSuffix( const Path : String; const PathSep : Char ) : String
8156: Procedure PathEnsureSuffixA( var Path : AnsiString; const PathSep : Char )
8157: Procedure PathEnsureSuffix( var Path : String; const PathSep : Char )
8158: Procedure PathEnsureNoSuffixA( var Path : AnsiString; const PathSep : Char )
8159: Procedure PathEnsureNoSuffix( var Path : String; const PathSep : Char )
8160: //Function PathCanonicalA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8161: Function PathCanonical( const Path : String; const PathSep : Char ) : String
8162: Function PathExpandA(const Path:AnsiString;const basePath:AnsiString;const PathSep:Char):AnsiString
8163: Function PathExpand( const Path : String; const basePath : String; const PathSep : Char ) : String
8164: Function PathLeftElementA( const Path : AnsiString; const PathSep : Char ) : AnsiString
8165: Function PathLeftElement( const Path : String; const PathSep : Char ) : String
8166: Procedure PathSplitLeftElementA( const Path:AString;var LeftElement,RightPath:AString;const PathSep:Char );
8167: Procedure PathSplitLeftElement( const Path:String; var LeftElement,RightPath: String;const PathSep:Char );
8168: Procedure DecodeFilePathA( const FilePath:AnsiString; var Path,FileName:AnsiString;const PathSep:Char );
8169: Procedure DecodeFilePath( const FilePath : String; var Path, FileName : String; const PathSep : Char )
8170: Function FileNameValidA( const FileName : AnsiString ) : AnsiString
8171: Function FileNameValid( const FileName : String ) : String
8172: Function FilePathA( const FileName,Path:AnsiString;const basePath:AnsiStr;const PathSep:Char ):AnsiString;
8173: Function FilePath( const FileName, Path: String;const basePath: String;const PathSep : Char ) : String
8174: Function DirectoryExpandA( const Path:AnsiString;const basePath:AnsiString;const PathSep:Char ):AnsiString
8175: Function DirectoryExpand( const Path : String; const basePath : String; const PathSep : Char ) : String
8176: Function UnixPathToWinPath( const Path : AnsiString ) : AnsiString
8177: Function WinPathToUnixPath( const Path : AnsiString ) : AnsiString
8178: Procedure CCopyFile( const FileName, DestName : String )
8179: Procedure CMoveFile( const FileName, DestName : String )
8180: Function CDeleteFiles( const FileMode : String ) : Boolean
8181: Function FileSeekEx( const FHandle:TFileHandle;const FileOffset:Int64; const FilePos:TFileSeekPos):Int64;
8182: Procedure FileCloseEx( const FileHandle : TFileHandle )
8183: Function FileExistsA( const FileName : AnsiString ) : Boolean
8184: Function CFileExists( const FileName : String ) : Boolean
8185: Function CFileGetSize( const FileName : String ) : Int64
8186: Function FileGetDateTime( const FileName : String ) : TDateTime
8187: Function FileGetDateTime2( const FileName : String ) : TDateTime
8188: Function FileIsReadOnly( const FileName : String ) : Boolean
8189: Procedure FileDeleteEx( const FileName : String )
8190: Procedure FileRenameEx( const OldFileName, NewFileName : String )
8191: Function ReadFileStrA( const FileName:AnsiString; const FileSharing : TFileSharing; const FileCreationMode : TFileCreationMode; const FileOpenWait : PFileOpenWait ) : AnsiString
8192: Function DirectoryEntryExists( const Name : String ) : Boolean
8193: Function DirectoryEntrySize( const Name : String ) : Int64
8194: Function CDirectoryExists( const DirectoryName : String ) : Boolean
8195: Function DirectoryGetDateTime( const DirectoryName : String ) : TDateTime
8196: Procedure CDirectoryCreate( const DirectoryName : String )
8197: Function GetFirstFileNameMatching( const FileMode : String ) : String
8198: Function DirEntryGetAttr( const FileName : AnsiString ) : Integer
8199: Function DirEntryIsDirectory( const FileName : AnsiString ) : Boolean
8200: Function FileHasAttr( const FileName : String; const Attr : Word ) : Boolean
8201:   AddTypeS('TLogicalDriveType','( DriveRemovable, DriveFixed, DriveRemote, '
8202:   +'DriveCDRom, DriveRamDisk, DriveTypeUnknown )'
8203: Function DriveIsValid( const Drive : Char ) : Boolean
8204: Function DriveGetType( const Path : AnsiString ) : TLogicalDriveType

```

```

8205: Function DriveFreeSpace( const Path : AnsiString ) : Int64
8206:
8207: procedure SIRegister_cTimers(CL: TPSPascalCompiler);
8208: begin
8209:   AddClassN(FindClass('TOBJECT'), 'ETimers'
8210:   Const ('TickFrequency', 'LongInt'( 1000));Function GetTick : LongWord
8211:   Function TickDelta( const D1, D2 : LongWord) : Integer
8212:   Function TickDeltaW( const D1, D2 : LongWord) : LongWord
8213:     AddTypeS('THPTimer', 'Int64'
8214:   Procedure StartTimer( var Timer : THPTimer)
8215:   Procedure StopTimer( var Timer : THPTimer)
8216:   Procedure ResumeTimer( var StoppedTimer : THPTimer)
8217:   Procedure InitStoppedTimer( var Timer : THPTimer)
8218:   Procedure InitElapsedTimer( var Timer : THPTimer; const Milliseconds : Integer)
8219:   Function MillisecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Integer
8220:   Function MicrosecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Int64
8221:   Procedure WaitMicroseconds( const MicroSeconds : Integer)
8222:   Function GetHighPrecisionFrequency : Int64
8223:   Function GetHighPrecisionTimerOverhead : Int64
8224:   Procedure AdjustTimerForOverhead( var StoppedTimer : THPTimer; const Overhead : Int64)
8225:   Procedure SelfTestCTimer
8226: end;
8227:
8228: procedure SIRegister_cRandom(CL: TPSPascalCompiler);
8229: begin
8230:   Function RandomSeed : LongWord
8231:   Procedure AddEntropy( const Value : LongWord)
8232:   Function RandomUniform : LongWord;
8233:   Function RandomUniform1( const N : Integer) : Integer;
8234:   Function RandomBoolean : Boolean
8235:   Function RandomByte : Byte
8236:   Function RandomByteNonZero : Byte
8237:   Function RandomWord : Word
8238:   Function RandomInt64 : Int64;
8239:   Function RandomInt641( const N : Int64) : Int64;
8240:   Function RandomHex( const Digits : Integer) : String
8241:   Function RandomFloat : Extended
8242:   Function RandomAlphaStr( const Length : Integer) : AnsiString
8243:   Function RandomPseudoWord( const Length : Integer) : AnsiString
8244:   Function RandomPassword(const MinL,MaxLength:Int;const CaseSens,UseSymbols,UseNumbers:Bool):AnsiString;
8245:   Function mwcRandomLongWord : LongWord
8246:   Function urnRandomLongWord : LongWord
8247:   Function moaRandomFloat : Extended
8248:   Function mwcRandomFloat : Extended
8249:   Function RandomNormalF : Extended
8250:   Procedure SelfTestCRandom
8251: end;
8252:
8253: procedure SIRegister_SynEditMiscProcs(CL: TPSPascalCompiler);
8254: begin
8255:   // PIntArray', 'TIntArray // will not work
8256:   Addtypes('TConvertTabsProc','function(const Line:AnsiString; TabWidth: integer):AnsiString
8257:   TConvertTabsProcEx, function(const Line:AnsiString;TabWidth:integer;var HasTabs:boolean):AnsiString
8258:   Function synMax( x, y : integer ) : integer
8259:   Function synMin( x, y : integer ) : integer
8260:   Function synMinMax( x, mi, ma : integer ) : integer
8261:   Procedure synSwapInt( var l, r : integer )
8262:   Function synMaxPoint( const P1, P2 : TPoint ) : TPoint
8263:   Function synMinPoint( const P1, P2 : TPoint ) : TPoint
8264:   //Function synGetIntArray( Count : Cardinal; InitialValue : integer ) : PIntArray
8265:   Procedure synInternalFillRect( dc : HDC; const rcPaint : TRect )
8266:   Function synGetBestConvertTabsProc( TabWidth : integer ) : TConvertTabsProc
8267:   Function synConvertTabs( const Line : AnsiString; TabWidth : integer ) : AnsiString
8268:   Function synGetBestConvertTabsProcEx( TabWidth : integer ) : TConvertTabsProcEx
8269:   Function synConvertTabsEx( const Line:AnsiString;TabWidth:integer; var HasTabs:boolean):AnsiString;
8270:   Function synGetExpandedLength( const aStr : string; aTabWidth : integer ) : integer
8271:   Function synCharIndex2CaretPos( Index, TabWidth : integer; const Line : string ) : integer
8272:   Function synCaretPos2CharIndex(Position,TabWidth:int;const Line:string;var InsideTabChar:boolean):int;
8273:   Function synStrScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8274:   Function synRScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8275:   TStringType', '( stNone, stHalfNumAlpha, stHalfSymbol, stHalfKat'
8276:     +'akana,stWideNumAlpha,stWideSymbol,stWideKatakana,stHiragana,stIdeograph,stControl,stKashida )
8277:   ('C3_NONSPACING','LongInt'( 1);
8278:   ('C3_DIACRITIC','LongInt'( 2);
8279:   ('C3_VOWELMARK','LongInt'( 4);
8280:   ('C3_SYMBOL','LongInt'( 8);
8281:   ('C3_KATAKANA','LongWord( $0010);
8282:   ('C3_HIRAGANA','LongWord( $0020);
8283:   ('C3_HALFWIDTH','LongWord( $0040);
8284:   ('C3_FULLWIDTH','LongWord( $0080);
8285:   ('C3_IDEOGRAPH','LongWord( $0100);
8286:   ('C3_KASHIDA','LongWord( $0200);
8287:   ('C3_LEXICAL','LongWord( $0400);
8288:   ('C3_ALPHA','LongWord( $8000);
8289:   ('C3_NOTAPPLICABLE','LongInt'( 0);
8290:   Function synStrScanForMultiByteChar( const Line : string; Start : Integer ) : Integer
8291:   Function synRScanForMultiByteChar( const Line : string; Start : Integer ) : Integer
8292:   Function synIsStringType( Value : Word ) : TStringType
8293:   Function synGetEOL( Line : PChar ) : PChar

```

```

8294: Function synEncodeString( s : string ) : string
8295: Function synDecodeString( s : string ) : string
8296: Procedure synFreeAndNil( var Obj: TObject )
8297: Procedure synAssert( Expr : Boolean )
8298: Function synLastDelimiter( const Delimiters, S : string ) : Integer
8299: TReplaceFlag', '( rfReplaceAll, rfIgnoreCase )
8300: TReplaceFlags', 'set of TReplaceFlag )
8301: Function synStringReplace(const S, OldPattern, NewPattern : string; Flags: TReplaceFlags) : string
8302: Function synGetRValue( RGBValue : TColor ) : byte
8303: Function synGetGValue( RGBValue : TColor ) : byte
8304: Function synGetBValue( RGBValue : TColor ) : byte
8305: Function synRGB( r, g, b : Byte ) : Cardinal
8306: // THighlighterAttriProc', 'Function ( Highlighter : TSynCustomHighlighter
8307: // +'lighter; Attris:TSynHighlighterAttributes;UniqueAttriName:string;Params array of Pointer):Boolean;
8308: //Function synEnumHighlighterAttrs( Highlighter : TSynCustomHighlighter; SkipDuplicates : Boolean;
HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer ) : Boolean
8309: Function synCalcFCS( const ABuf, ABufSize : Cardinal ) : Word
8310: Procedure synSynDrawGradient(const ACanvas:TCanvas;const AStartColor,
AEndColor:TColor;ASteps:integer;const ARect : TRect; const AHorizontal : boolean)
8311: end;
8312:
8313: Function GET_APPCOMMAND_LPARAM( lParam : LPARAM ) : WORD
8314: Function GET_DEVICE_LPARAM( lParam : LPARAM ) : WORD
8315: Function GET_KEYSTATE_LPARAM( lParam : LPARAM ) : WORD
8316:
8317: procedure SIRegister_synautil(CL: TPSPascalCompiler);
8318: begin
8319:   Function STimeZoneBias : integer
8320:   Function TimeZone : string
8321:   Function Rfc822DateDateTime( t : TDateTime ) : string
8322:   Function CDateTime( t : TDateTime ) : string
8323:   Function SimpleDateTime( t : TDateTime ) : string
8324:   Function AnsiDateTime( t : TDateTime ) : string
8325:   Function GetMonthNumber( Value : String ) : integer
8326:   Function GettimeFromStr( Value : string ) : TDateTime
8327:   Function GetDateMDYFromStr( Value : string ) : TDateTime
8328:   Function DecodeRFCDateTime( Value : string ) : TDateTime
8329:   Function GetUTTTime : TDateTime
8330:   Function SetUTTTime( Newdt : TDateTime ) : Boolean
8331:   Function SGetTick : LongWord
8332:   Function SStickDelta( TickOld, TickNew : LongWord ) : LongWord
8333:   Function CodeInt( Value : Word ) : Ansistring
8334:   Function DecodeInt( const Value : Ansistring; Index : Integer ) : Word
8335:   Function CodeLongInt( Value : LongInt ) : Ansistring
8336:   Function DecodeLongInt( const Value : Ansistring; Index : Integer ) : LongInt
8337:   Function DumpStr( const Buffer : Ansistring ) : string
8338:   Function DumpExStr( const Buffer : Ansistring ) : string
8339:   Procedure Dump( const Buffer : AnsiString; DumpFile : string )
8340:   Procedure DumpEx( const Buffer : AnsiString; DumpFile : string )
8341:   Function TrimSPLeft( const S : string ) : string
8342:   Function TrimSPRight( const S : string ) : string
8343:   Function TrimSP( const S : string ) : string
8344:   Function SeparateLeft( const Value, Delimiter : string ) : string
8345:   Function SeparateRight( const Value, Delimiter : string ) : string
8346:   Function SGetParameter( const Value, Parameter : string ) : string
8347:   Procedure ParseParametersEx( Value, Delimiter : string; const Parameters : TStrings )
8348:   Procedure ParseParameters( Value : string; const Parameters : TStrings )
8349:   Function IndexByBegin( Value : string; const List : TStrings ) : integer
8350:   Function GetEmailAddr( const Value : string ) : string
8351:   Function GetEmailDesc( Value : string ) : string
8352:   Function CStrToHex( const Value : Ansistring ) : string
8353:   Function CIntToBin( Value : Integer; Digits : Byte ) : string
8354:   Function CBinToInt( const Value : string ) : Integer
8355:   Function ParseURL( URL : string; var Prot, User, Pass, Host, Port, Path, Para:string):string
8356:   Function CReplaceString( Value, Search, Replace : AnsiString ) : AnsiString
8357:   Function CRPosEx( const Sub, Value : string; From : integer ) : Integer
8358:   Function CRPos( const Sub, Value : String ) : Integer
8359:   Function FetchBin( var Value : string; const Delimiter : string ) : string
8360:   Function CFetch( var Value : string; const Delimiter : string ) : string
8361:   Function FetchEx( var Value : string; const Delimiter, Quotation : string ) : string
8362:   Function IsBinaryString( const Value : AnsiString ) : Boolean
8363:   Function PosCRLF( const Value : AnsiString; var Terminator : AnsiString ) : integer
8364:   Procedure StringsTrim( const value : TStrings )
8365:   Function PosFrom( const SubStr, Value : String; From : integer ) : integer
8366:   Function IncPoint( const p : __pointer; Value : integer ) : __pointer
8367:   Function GetBetween( const PairBegin, PairEnd, Value : string ) : string
8368:   Function CCountOfChar( const Value : string; aChr : char ) : integer
8369:   Function UnquoteStr( const Value : string; Quote : Char ) : string
8370:   Function QuoteStr( const Value : string; Quote : Char ) : string
8371:   Procedure HeadersToList( const Value : TStrings )
8372:   Procedure ListToHeaders( const Value : TStrings )
8373:   Function SwapBytes( Value : integer ) : integer
8374:   Function ReadStrFromStream( const Stream : TStream; len : integer ) : AnsiString
8375:   Procedure WriteStrToStream( const Stream : TStream; Value : AnsiString )
8376:   Function GetTempFile( const Dir, prefix : AnsiString ) : AnsiString
8377:   Function CPadString( const Value : AnsiString; len : integer; Pad : AnsiChar ) : AnsiString
8378:   Function CXorString( Indata1, Indata2 : AnsiString ) : AnsiString
8379:   Function NormalizeHeader( Value : TStrings; var Index : Integer ) : string
8380: end;

```

```

8381:
8382: procedure SIRegister_StCRC(CL: TPSPascalCompiler);
8383: begin
8384:   ('CrcBufSize', 'LongInt' ( 2048));
8385:   Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8386:   Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8387:   Function Adler32OfFile( FileName : AnsiString) : LongInt
8388:   Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8389:   Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8390:   Function Crc16OfFile( FileName : AnsiString) : Cardinal
8391:   Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8392:   Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8393:   Function Crc32OfFile( FileName : AnsiString) : LongInt
8394:   Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8395:   Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8396:   Function InternetSumOfFile( FileName : AnsiString) : Cardinal
8397:   Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8398:   Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8399:   Function Kermit16OfFile( FileName : AnsiString) : Cardinal
8400: end;
8401:
8402: procedure SIRegister_ComObj(cl: TPSPascalCompiler);
8403: begin
8404:   function CreateOleObject(const ClassName: String): IDispatch;
8405:   function GetActiveOleObject(const ClassName: String): IDispatch;
8406:   function ProgIDToClassID(const ProgID: string): TGUID;
8407:   function ClassIDToProgID(const ClassID: TGUID): string;
8408:   function CreateClassID: string;
8409:   function CreateGUIDString: string;
8410:   function CreateGUIDID: string;
8411:   procedure OleError(ErrorCode: longint);
8412:   procedure OleCheck(Result: HResult);
8413: end;
8414:
8415:   Function xCreateOleObject( const ClassName : string ) : Variant //or IDispatch
8416:   Function xGetActiveOleObject( const ClassName : string ) : Variant
8417: //Function DllGetClassObject( const CLSID : TGUID; const IID : TIID; var Obj ) : HResult
8418:   Function DllCanUnloadNow : HResult
8419:   Function DllRegisterServer : HResult
8420:   Function DllUnregisterServer : HResult
8421:   Function VarFromInterface( Unknown : IUnknown ) : Variant
8422:   Function VarToInterface( const V : Variant ) : IDispatch
8423:   Function VarToAutoObject( const V : Variant ) : TAutoObject
8424: //Procedure
8425: DispInvoke(Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer;Res:PVariant);
8426: //Procedure DispInvokeError( Status : HResult; const ExcepInfo : TExcepInfo)
8427: Procedure OleError( ErrorCode : HResult )
8428: Procedure OleCheck( Result : HResult )
8429: Function StringToClassID( const S : string ) : TGUID
8430: Function ClassIDToString( const ClassID : TGUID ) : string
8431: Function xProgIDToClassID( const ProgID : string ) : TGUID
8432: Function xClassIDToProgID( const ClassID : TGUID ) : string
8433: Function xWideCompareStr( const S1, S2 : WideString ) : Integer
8434: Function xWideSameStr( const S1, S2 : WideString ) : Boolean
8435: Function xGUIDToString( const ClassID : TGUID ) : string
8436: Function xStringToGUID( const S : string ) : TGUID
8437: Function xGetModuleName( Module : HMODULE ) : string
8438: Function xIfThen( AValue : Boolean; const ATrue : Integer; const AFalse : Integer ) : Integer
8439: Function xUtf8Encode( const WS : WideString ) : UTF8String
8440: Function xUtf8Decode( const S : UTF8String ) : WideString
8441: Function xExcludeTrailingPathDelimiter( const S : string ) : string
8442: Function xIncludeTrailingPathDelimiter( const S : string ) : string
8443: Function XRTLHandleCOMException : HResult
8444: Procedure XRTLCheckArgument( Flag : Boolean )
8445: //Procedure XRTLCheckOutArgument( out Arg )
8446: Procedure XRTLInterfaceConnect( const Source:IUnknown;const IID:TIID;const Sink:IUnknown;var Connection:longint );
8447: Procedure XRTLInterfaceDisconnect( const Source: IUnknown; const IID:TIID;var Connection : Longint )
8448: Function XRTLRegisterActiveObject( const Unk:IUnknown;ClassID:TGUID;Flags:DWORD;var RegisterCookie:Int ):HResult
8449: Function XRTLUUnRegisterActiveObject( var RegisterCookie : Integer ) : HResult
8450: //Function XRTLGetActiveObject( ClassID : TGUID; RIID : TIID; out Obj ) : HResult
8451: Procedure XRTLEnumActiveObjects( Strings : TStrings );
8452: function XRTLDefaultCategoryManager: IUnknown;
8453: function XRTLIsCategoryEmpty(CatID: TGUID; const CategoryManager: IUnknown = nil): Boolean;
8454: // ICatRegister helper functions
8455: function XRTLCreateComponentCategory(CatID: TGUID; CatDescription: WideString;
8456:                                         LocaleID: TGUID = LOCALE_USER_DEFAULT;
8457:                                         const CategoryManager: IUnknown = nil): HResult;
8458: function XRTLRemoveComponentCategory(CatID: TGUID; CatDescription: WideString;
8459:                                         LocaleID: TGUID = LOCALE_USER_DEFAULT;
8460:                                         const CategoryManager: IUnknown = nil): HResult;
8461: function XRTLRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8462:                                         const CategoryManager: IUnknown = nil): HResult;
8463: function XRTLUUnRegisterCLSIDInCategory(CatID: TGUID; CatID: TGUID;
8464:                                         const CategoryManager: IUnknown = nil): HResult;
8465: // ICatInformation helper functions
8466: function XRTLGetCategoryDescription(CatID: TGUID; var CatDescription: WideString;

```

```

8467:                               LocaleID: TLCID = LOCALE_USER_DEFAULT;
8468:                               const CategoryManager: IUnknown = nil): HResult;
8469: function XRTLGetCategoryList(Strings: TStrings; LocaleID: TLCID = LOCALE_USER_DEFAULT;
8470:                               const CategoryManager: IUnknown = nil): HResult;
8471: function XRTLGetCategoryCLSIDList(CATID: TGUID; Strings: TStrings;
8472:                               const CategoryManager: IUnknown = nil): HResult;
8473: function XRTLGetCategoryProgIDList(CATID: TGUID; Strings: TStrings;
8474:                               const CategoryManager: IUnknown = nil): HResult;
8475: function XRTLFetch(var AInput: WideString; const ADelim: WideString = ' ';
8476:                               const ADelete: Boolean = True): WideString;
8477: function XRTLPos(const ASub, AIn: WideString; AStart: Integer = -1): Integer;
8478: Function XRTLGetVariantAsString( const Value : Variant ) : string
8479: Function XRTLDeltaTimeToTimeZoneTime( DT : TDateTime; TimeZone : TXRTLTimeZone ) : TDateTime
8480: Function XRTLGetTimeZones : TXRTLTimeZones
8481: Function XFileTimeToDate( FileTime : TFileTime ) : TDateTime
8482: Function DateToFileName( Date : TDateTime ) : TFileName
8483: Function GMTNow : TDateTime
8484: Function GMTToLocalTime( GMT : TDateTime ) : TDateTime
8485: Function LocalTimeToGMT( LocalTime : TDateTime ) : TDateTime
8486: Procedure XRTLNotImplemented
8487: Procedure XTRTLRaiseError( E : Exception )
8488: Procedure XRTLInvalidOperation( ClassName:string; OperationName:string; Description: string )
8489:
8490:
8491: procedure SIRegister_xrtl_util_Value(CL: TPPascalCompiler);
8492: begin
8493:   SIRegister_IXRTLValue(CL);
8494:   SIRegister_TXRTLValue(CL);
8495:   //AddTypeS('PXRTLValueArray', '^TXRTLValueArray // will not work
8496:   AddTypes('TXRTLValueArray', 'array of IXRTLValue
8497:   Function XRTLValue( const AValue : Cardinal ) : IXRTLValue;
8498:   Function XRTLSetValue( const IValue : IXRTLValue; const AValue : Cardinal ) : Cardinal;
8499:   Function XRTLGetAsCardinal( const IValue : IXRTLValue ) : Cardinal;
8500:   Function XRTLGetAsCardinalDef( const IValue : IXRTLValue; const DefValue : Cardinal ) : Cardinal;
8501:   Function XRTLValue1( const AValue : Integer ) : IXRTLValue;
8502:   Function XRTLSetValue1( const IValue : IXRTLValue; const AValue : Integer ) : Integer;
8503:   Function XRTLGetAsInteger( const IValue : IXRTLValue ) : Integer;
8504:   Function XRTLGetAsIntegerDef( const IValue : IXRTLValue; const DefValue : Integer ) : Integer;
8505:   Function XRTLValue2( const AValue : Int64 ) : IXRTLValue;
8506:   Function XRTLSetValue2( const IValue : IXRTLValue; const AValue : Int64 ) : Int64;
8507:   Function XRTLGetAsInt64( const IValue : IXRTLValue ) : Int64;
8508:   Function XRTLGetAsInt64Def( const IValue : IXRTLValue; const DefValue : Int64 ) : Int64;
8509:   Function XRTLValue3( const AValue : Single ) : IXRTLValue;
8510:   Function XRTLSetValue3( const IValue : IXRTLValue; const AValue : Single ) : Single;
8511:   Function XRTLGetAsSingle( const IValue : IXRTLValue ) : Single;
8512:   Function XRTLGetAsSingleDef( const IValue : IXRTLValue; const DefValue : Single ) : Single;
8513:   Function XRTLValue4( const AValue : Double ) : IXRTLValue;
8514:   Function XRTLSetValue4( const IValue : IXRTLValue; const AValue : Double ) : Double;
8515:   Function XRTLGetAsDouble( const IValue : IXRTLValue ) : Double;
8516:   Function XRTLGetAsDoubleDef( const IValue : IXRTLValue; const DefValue : Double ) : Double;
8517:   Function XRTLValue5( const AValue : Extended ) : IXRTLValue;
8518:   Function XRTLSetValue5( const IValue : IXRTLValue; const AValue : Extended ) : Extended;
8519:   Function XRTLGetAsExtended( const IValue : IXRTLValue ) : Extended;
8520:   Function XRTLGetAsExtendedDef( const IValue : IXRTLValue; const DefValue : Extended ) : Extended;
8521:   Function XRTLValue6( const AValue : IInterface ) : IXRTLValue;
8522:   Function XRTLSetValue6( const IValue : IXRTLValue; const AValue : IInterface ) : IInterface;
8523:   Function XRTLGetAsInterface( const IValue : IXRTLValue ) : IInterface;
8524:   //Function XRTLGetAsInterface( const IValue : IXRTLValue; out Obj ) : IInterface;
8525:   Function XRTLGetAsInterfaceDef( const IValue : IXRTLValue; const DefValue : IInterface ) : IInterface;
8526:   Function XRTLValue7( const AValue : WideString ) : IXRTLValue;
8527:   Function XRTLSetValue7( const IValue : IXRTLValue; const AValue : WideString ) : WideString;
8528:   Function XRTLGetAsWideString( const IValue : IXRTLValue ) : WideString;
8529:   Function XRTLGetAsWideStringDef( const IValue : IXRTLValue; const DefValue : WideString ) : WideString;
8530:   Function XRTLValue8( const AValue : TObject; const AOwnValue : Boolean ) : IXRTLValue;
8531:   Function XRTLSetValue8( const IValue : IXRTLValue; const AValue : TObject ) : TObject;
8532:   Function XRTLGetAsObject( const IValue : IXRTLValue; const ADetachOwnership : Boolean ) : TObject;
8533:   Function XRTLGetAsObjectDef( const IValue : IXRTLValue; const DefValue : TObject; const
8534:     ADetachOwnership : Boolean ) : TObject;
8535:   //Function XRTLSetValue9( const IValue : IXRTLValue; const AValue : _Pointer ) : _Pointer;
8536:   //Function XRTLGetAsPointer( const IValue : IXRTLValue ) : _Pointer;
8537:   //Function XRTLGetAsPointerDef( const IValue : IXRTLValue; const DefValue : _Pointer ) : _Pointer;
8538:   Function XRTLValueV( const AValue : Variant ) : IXRTLValue;
8539:   Function XRTLSetValueV( const IValue : IXRTLValue; const AValue : Variant ) : Variant;
8540:   Function XRTLGetAsVariant( const IValue : IXRTLValue ) : Variant;
8541:   Function XRTLGetAsVariantDef( const IValue : IXRTLValue; const DefValue : Variant ) : Variant;
8542:   Function XRTLValue10( const AValue : Currency ) : IXRTLValue;
8543:   Function XRTLSetValue10( const IValue : IXRTLValue; const AValue : Currency ) : Currency;
8544:   Function XRTLGetAsCurrency( const IValue : IXRTLValue ) : Currency;
8545:   Function XRTLGetAsCurrencyDef( const IValue : IXRTLValue; const DefValue : Currency ) : Currency;
8546:   Function XRTLValue11( const AValue : Comp ) : IXRTLValue;
8547:   Function XRTLSetValue11( const IValue : IXRTLValue; const AValue : Comp ) : Comp;
8548:   Function XRTLGetAsComp( const IValue : IXRTLValue ) : Comp;
8549:   Function XRTLGetAsCompDef( const IValue : IXRTLValue; const DefValue : Comp ) : Comp;
8550:   Function XRTLValue12( const AValue : TClass ) : IXRTLValue;
8551:   Function XRTLSetValue12( const IValue : IXRTLValue; const AValue : TClass ) : TClass;
8552:   Function XRTLGetAsClass( const IValue : IXRTLValue ) : TClass;
8553:   Function XRTLGetAsClassDef( const IValue : IXRTLValue; const DefValue : TClass ) : TClass;
8554:   Function XRTLValue13( const AValue : TGUID ) : IXRTLValue;

```

```

8555: Function XRTLSetValue13( const IValue : IXRTLValue; const AValue : TGUID ) : TGUID;
8556: Function XRTLGetAsGUID( const IValue : IXRTLValue ) : TGUID
8557: Function XRTLGetAsGUIDDef( const IValue : IXRTLValue; const DefValue : TGUID ) : TGUID
8558: Function XRTLValue14( const AValue : Boolean ) : IXRTLValue;
8559: Function XRTLSetValue14( const IValue : IXRTLValue; const AValue : Boolean ) : Boolean;
8560: Function XRTLGetAsBoolean( const IValue : IXRTLValue ) : Boolean
8561: Function XRTLGetAsBooleanDef( const IValue : IXRTLValue; const DefValue : Boolean ) : Boolean
8562: end;
8563:
8564: //*****unit uPSI_GR32;*****
8565:
8566: Function Color32( WinColor : TColor ) : TColor32;
8567: Function Color321( R, G, B : Byte; A : Byte ) : TColor32;
8568: Function Color322( Index : Byte; var Palette : TPalette32 ) : TColor32;
8569: Function Gray32( Intensity : Byte; Alpha : Byte ) : TColor32;
8570: Function WinColor( Color32 : TColor32 ) : TColor;
8571: Function ArrayOfColor32( Colors : array of TColor32 ) : TArrayOfColor32;
8572: Procedure Color32ToRGB( Color32 : TColor32; var R, G, B : Byte );
8573: Procedure Color32ToRGB( Color32 : TColor32; var R, G, B, A : Byte );
8574: Function Color32Components( R, G, B, A : Boolean ) : TColor32Components;
8575: Function RedComponent( Color32 : TColor32 ) : Integer;
8576: Function GreenComponent( Color32 : TColor32 ) : Integer;
8577: Function BlueComponent( Color32 : TColor32 ) : Integer;
8578: Function AlphaComponent( Color32 : TColor32 ) : Integer;
8579: Function Intensity( Color32 : TColor32 ) : Integer;
8580: Function SetAlpha( Color32 : TColor32; NewAlpha : Integer ) : TColor32;
8581: Function HSLtoRGB( H, S, L : Single ) : TColor32;
8582: Procedure RGBtoHSL( RGB : TColor32; out H, S, L : Single );
8583: Function HSLtoRGB( H, S, L : Integer ) : TColor32;
8584: Procedure RGBtoHSL1( RGB : TColor32; out H, S, L : Byte );
8585: Function WinPalette( const P : TPalette32 ) : HPALETTE;
8586: Function FloatPoint( X, Y : Single ) : TFloatPoint;
8587: Function FloatPoint1( const P : TPoint ) : TFloatPoint;
8588: Function FloatPoint2( const FXP : TFixedPoint ) : TFloatPoint;
8589: Function FixedPoint( X, Y : Integer ) : TFixedPoint;
8590: Function FixedPoint1( X, Y : Single ) : TFixedPoint;
8591: Function FixedPoint2( const P : TPoint ) : TFixedPoint;
8592: Function FixedPoint3( const FP : TFloatPoint ) : TFixedPoint;
8593: AddTypes('TRectRounding', '( rrClosest, rrOutside, rrInside )');
8594: Function MakeRect( const L, T, R, B : Integer ) : TRect;
8595: Function MakeRect1( const FR : TFloatRect; Rounding : TRectRounding ) : TRect;
8596: Function MakeRect2( const FXR : TRect; Rounding : TRectRounding ) : TRect;
8597: Function GFFixedRect( const L, T, R, B : TFixed ) : TRect;
8598: Function FixedRect1( const ARect : TRect ) : TRect;
8599: Function FixedRect2( const FR : TFloatRect ) : TRect;
8600: Function GFFloatRect( const L, T, R, B : TFloat ) : TFloatRect;
8601: Function FloatRect1( const ARect : TRect ) : TFloatRect;
8602: Function FloatRect2( const FXR : TRect ) : TFloatRect;
8603: Function GIIntersectRect( out Dst : TRect; const R1, R2 : TRect ) : Boolean;
8604: Function IntersectRect1( out Dst : TFloatRect; const FR1, FR2 : TFloatRect ) : Boolean;
8605: Function GUUnionRect( out Rect : TRect; const R1, R2 : TRect ) : Boolean;
8606: Function UnionRect1( out Rect : TFloatRect; const R1, R2 : TFloatRect ) : Boolean;
8607: Function GEEqualRect( const R1, R2 : TRect ) : Boolean;
8608: Function EqualRect1( const R1, R2 : TFloatRect ) : Boolean;
8609: Procedure GIInflateRect( var R : TRect; Dx, Dy : Integer );
8610: Procedure InflateRect1( var FR : TFloatRect; Dx, Dy : TFloat );
8611: Procedure GOffsetRect( var R : TRect; Dx, Dy : Integer );
8612: Procedure OffsetRect1( var FR : TFloatRect; Dx, Dy : TFloat );
8613: Function IsRectEmpty( const R : TRect ) : Boolean;
8614: Function IsRectEmpty1( const FR : TFloatRect ) : Boolean;
8615: Function GPtInRect( const R : TRect; const P : TPoint ) : Boolean;
8616: Function PtInRect1( const R : TFloatRect; const P : TPoint ) : Boolean;
8617: Function PtInRect2( const R : TRect; const P : TFloatPoint ) : Boolean;
8618: Function PtInRect3( const R : TFloatRect; const P : TFloatPoint ) : Boolean;
8619: Function EqualRectSize( const R1, R2 : TRect ) : Boolean;
8620: Function EqualRectSize1( const R1, R2 : TFloatRect ) : Boolean;
8621: Function MessageBeep( uType : UINT ) : BOOL;
8622: Function ShowCursor( bShow : BOOL ) : Integer;
8623: Function SetCursorPos( X, Y : Integer ) : BOOL;
8624: Function SetCursor( hCursor : HICON ) : HCURSOR;
8625: Function GetCursorPos( var lpPoint : TPoint ) : BOOL;
8626: //Function ClipCursor( lpRect : PRect ) : BOOL;
8627: Function GetClipCursor( var lpRect : TRect ) : BOOL;
8628: Function GetCursor : HCURSOR;
8629: Function CreateCaret( hWnd : HWND; hBitmap : HBITMAP; nWidth, nHeight : Integer ) : BOOL;
8630: Function GetCaretBlinkTime : UINT;
8631: Function SetCaretBlinkTime( uMSeconds : UINT ) : BOOL;
8632: Function DestroyCaret : BOOL;
8633: Function HideCaret( hWnd : HWND ) : BOOL;
8634: Function ShowCaret( hWnd : HWND ) : BOOL;
8635: Function SetCaretPos( X, Y : Integer ) : BOOL;
8636: Function GetCaretPos( var lpPoint : TPoint ) : BOOL;
8637: Function ClientToScreen( hWnd : HWND; var lpPoint : TPoint ) : BOOL;
8638: Function ScreenToClient( hWnd : HWND; var lpPoint : TPoint ) : BOOL;
8639: Function MapWindowPoints( hWndFrom, hWndTo : HWND; var lpPoints, cPoints : UINT ) : Integer;
8640: Function WindowFromPoint( Point : TPoint ) : HWND;
8641: Function ChildWindowFromPoint( hWndParent : HWND; Point : TPoint ) : HWND;
8642:
8643:

```

```

8644: procedure SIRegister_GR32_Math(CL: TPSPPascalCompiler);
8645: begin
8646:   Function FixedFloor( A : TFixed ) : Integer
8647:   Function FixedCeil( A : TFixed ) : Integer
8648:   Function FixedMul( A, B : TFixed ) : TFixed
8649:   Function FixedDiv( A, B : TFixed ) : TFixed
8650:   Function OneOver( Value : TFixed ) : TFixed
8651:   Function FixedRound( A : TFixed ) : Integer
8652:   Function FixedSqr( Value : TFixed ) : TFixed
8653:   Function FixedSqrtLP( Value : TFixed ) : TFixed
8654:   Function FixedSqrtHP( Value : TFixed ) : TFixed
8655:   Function FixedCombine( W, X, Y : TFixed ) : TFixed
8656:   Procedure GRSinCos( const Theta : TFloat; out Sin, Cos : TFloat );
8657:   Procedure GRSinCos1( const Theta, Radius : Single; out Sin, Cos : Single );
8658:   Function GRHypot( const X, Y : TFloat ) : TFloat;
8659:   Function Hypotl( const X, Y : Integer ) : Integer;
8660:   Function FastSqrt( const Value : TFloat ) : TFloat
8661:   Function FastSqrtBab1( const Value : TFloat ) : TFloat
8662:   Function FastSqrtBab2( const Value : TFloat ) : TFloat
8663:   Function FastInvSqrt( const Value : Single ) : Single;
8664:   Function MulDiv( Multiplicand, Multiplier, Divisor : Integer ) : Integer
8665:   Function GRIIsPowerOf2( Value : Integer ) : Boolean
8666:   Function PrevPowerOf2( Value : Integer ) : Integer
8667:   Function NextPowerOf2( Value : Integer ) : Integer
8668:   Function Average( A, B : Integer ) : Integer
8669:   Function GRSign( Value : Integer ) : Integer
8670:   Function FloatMod( x, y : Double ) : Double
8671: end;
8672:
8673: procedure SIRegister_GR32_LowLevel(CL: TPSPPascalCompiler);
8674: begin
8675:   Function Clamp( const Value : Integer ) : Integer;
8676:   Procedure GRFillWord( var X, Count : Cardinal; Value : Longword )
8677:   Function StackAlloc( Size : Integer ) : Pointer
8678:   Procedure StackFree( P : Pointer )
8679:   Procedure Swap( var A, B : Pointer );
8680:   Procedure Swap1( var A, B : Integer );
8681:   Procedure Swap2( var A, B : TFixed );
8682:   Procedure Swap3( var A, B : TColor32 );
8683:   Procedure TestSwap( var A, B : Integer );
8684:   Procedure TestSwap1( var A, B : TFixed );
8685:   Function TestClip( var A, B : Integer; const Size : Integer ) : Boolean;
8686:   Function TestClip1( var A, B : Integer; const Start, Stop : Integer ) : Boolean;
8687:   Function GRConstrain( const Value, Lo, Hi : Integer ) : Integer;
8688:   Function Constrain1( const Value, Lo, Hi : Single ) : Single;
8689:   Function SwapConstrain( const Value:Integer; Constrain1,Constrain2:Integer ) : Integer
8690:   Function GRMin( const A, B, C : Integer ) : Integer;
8691:   Function GRMax( const A, B, C : Integer ) : Integer;
8692:   Function Clamp( Value, Max : Integer ) : Integer;
8693:   Function Clamp1( Value, Min, Max : Integer ) : Integer;
8694:   Function Wrap( Value, Max : Integer ) : Integer;
8695:   Function Wrap1( Value, Min, Max : Integer ) : Integer;
8696:   Function Wrap3( Value, Max : Single ) : Single;;
8697:   Function WrapPow2( Value, Max : Integer ) : Integer;
8698:   Function WrapPow21( Value, Min, Max : Integer ) : Integer;
8699:   Function Mirror( Value, Max : Integer ) : Integer;
8700:   Function Mirror1( Value, Min, Max : Integer ) : Integer;
8701:   Function MirrorPow2( Value, Max : Integer ) : Integer;
8702:   Function MirrorPow21( Value, Min, Max : Integer ) : Integer;
8703:   Function GetOptimalWrap( Max : Integer ) : TWrapProc;
8704:   Function GetOptimalWrap1( Min, Max : Integer ) : TWrapProcEx;
8705:   Function GetOptimalMirror( Max : Integer ) : TWrapProc;
8706:   Function GetOptimalMirror1( Min, Max : Integer ) : TWrapProcEx;
8707:   Function GetWrapProc( WrapMode : TWrapMode ) : TWrapProc;
8708:   Function GetWrapProc1( WrapMode : TWrapMode; Max : Integer ) : TWrapProc;
8709:   Function GetWrapProcEx( WrapMode : TWrapMode ) : TWrapProcEx;
8710:   Function GetWrapProcEx1( WrapMode : TWrapMode; Min, Max : Integer ) : TWrapProcEx;
8711:   Function Div255( Value : Cardinal ) : Cardinal
8712:   Function SAR_4( Value : Integer ) : Integer
8713:   Function SAR_8( Value : Integer ) : Integer
8714:   Function SAR_9( Value : Integer ) : Integer
8715:   Function SAR_11( Value : Integer ) : Integer
8716:   Function SAR_12( Value : Integer ) : Integer
8717:   Function SAR_13( Value : Integer ) : Integer
8718:   Function SAR_14( Value : Integer ) : Integer
8719:   Function SAR_15( Value : Integer ) : Integer
8720:   Function SAR_16( Value : Integer ) : Integer
8721:   Function ColorSwap( WinColor : TColor ) : TColor32
8722: end;
8723:
8724: procedure SIRegister_GR32_Filters(CL: TPSPPascalCompiler);
8725: begin
8726:   AddTypeS('TLogicalOperator', '( loXOR, loAND, loOR )
8727:   Procedure CopyComponents( Dst, Src : TCustomBitmap32; Components : TColor32Components );
8728:   Procedure CopyComponents1(Dst:TCustBmap32;DstX,
8729:     DstY:Int;Src:TCustomBmap32;SrcRect:TRect;Components:TColor32Comp;
8730:   Procedure AlphaToGrayscale( Dst, Src : TCustomBitmap32 );
8731:   Procedure ColorToGrayscale( Dst, Src : TCustomBitmap32; PreserveAlpha : Boolean );
8732:   Procedure IntensityToAlpha( Dst, Src : TCustomBitmap32 );

```

```

8732: Procedure Invert( Dst, Src : TCustomBitmap32; Components : TColor32Components )
8733: Procedure InvertRGB( Dst, Src : TCustomBitmap32 )
8734: Procedure ApplyLUT( Dst, Src : TCustomBitmap32; const LUT : TLUT8; PreserveAlpha : Boolean )
8735: Procedure ChromaKey( ABitmap : TCustomBitmap32; TrColor : TColor32 )
8736: Function CreateBitmap( Components : TColor32Components ) : TColor32
8737: Procedure ApplyBitmask(Dst : TCustomBitmap32; DstX,DstY:Integer; Src:TCustomBitmap32; SrcRect : TRect;
  Bitmask : TColor32; LogicalOperator : TLogicalOperator );
8738: Procedure
  ApplyBitmask1(ABitmap:TCustomBitmap32;ARect:TRect;Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8739: Procedure CheckParams( Dst, Src : TCustomBitmap32; ResizeDst : Boolean )
8740: end;
8741:
8742:
8743: procedure SIRegister_JclNTFS(CL: TPSPascalCompiler);
8744: begin
8745:   AddClassN(FindClass ('TOBJECT'), 'EJclNtfsError
8746:   AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )
8747:   Function NtfsGetCompression( const FileName : string; var State : Short ) : Boolean;
8748:   Function NtfsGetCompression1( const FileName : string ) : TFileCompressionState;
8749:   Function NtfsSetCompression( const FileName : string; const State : Short ) : Boolean
8750:   Procedure NtfsSetFileCompression( const FileName : string; const State : TFileCompressionState )
8751:   Procedure NtfsSetDirectoryTreeCompression(const Directory: string; const State : TFileCompressionState)
8752:   Procedure NtfsSetDefaultFileCompression(const Directory: string; const State:TFileCompressionState)
8753:   Procedure NtfsSetPathCompression(const Path:string;const State:TFileCompressionState;Recursive:Boolean;
8754:   //AddTypeS('TNtfsAllocRanges', 'record Entries : Integer; Data : PFileAlloca'
8755:   //'+tedRangeBuffer; MoreData : Boolean; end
8756:   Function NtfsSetSparse( const FileName : string ) : Boolean
8757:   Function NtfsZeroDataByHandle( const Handle : THandle; const First, Last : Int64 ) : Boolean
8758:   Function NtfsZeroDataByName( const FileName : string; const First, Last : Int64 ) : Boolean
8759:   //Function NtfsQueryAllocRanges(const FileName:string;Offset,Count:Int64;var
8760:   //Function NtfsGetAllocRangeEntry( const Ranges : TNtfsAllocRanges;
8761:   Index:Integer):TFileAllocatedRangeBuffer
8762:   Function NtfsSparseStreamsSupported( const Volume : string ) : Boolean
8763:   Function NtfsGetSparse( const FileName : string ) : Boolean
8764:   Function NtfsDeleteReparsePoint( const FileName : string; ReparseTag : DWORD ) : Boolean
8765:   Function NtfsSetReparsePoint( const FileName : string; var ReparseData, Size : Longword ) : Boolean
8766:   //Function NtfsGetReparsePoint(const FileName:string; var ReparseData:TReparseGuidDataBuffer):Boolean
8767:   Function NtfsGetReparseTag( const Path : string; var Tag : DWORD ) : Boolean
8768:   Function NtfsReparsePointsSupported( const Volume : string ) : Boolean
8769:   Function NtfsFileHasReparsePoint( const Path : string ) : Boolean
8770:   Function NtfsIsFolderMountPoint( const Path : string ) : Boolean
8771:   Function NtfsMountVolume( const Volume : Char; const MountPoint : string ) : Boolean
8772:   AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )
8773:   Function NtfsOpLockAckClosePending( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8774:   Function NtfsOpLockBreakAckNo2( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8775:   Function NtfsOpLockBreakAcknowledge( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8776:   Function NtfsOpLockBreakNotify( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8777:   Function NtfsRequestOpLock( Handle : THandle; Kind : TopLock; Overlapped : TOverlapped ) : Boolean
8778:   Function NtfsCreateJunctionPoint( const Source, Destination : string ) : Boolean
8779:   Function NtfsDeleteJunctionPoint( const Source : string ) : Boolean
8780:   Function NtfsGetJunctionPointDestination( const Source : string; var Destination : string ) : Boolean
8781:   AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
8782:   +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile )
8783:   AddTypeS('TStreamIds', 'set of TStreamId
8784:   AddTypeS('TInternalFindStreamData', 'record FileHandle : THandle; Context :
8785:   +': __Pointer; StreamIds : TStreamIds; end
8786:   AddTypeS('TFindStreamData', 'record internal : TInternalFindStreamData; At'
8787:   +'tributes : DWORD; StreamID : TStreamId; Name : WideString; Size : Int64; end
8788:   Function NtfsFindFirstStream(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Boolean;
8789:   Function NtfsFindNextStream( var Data : TFindStreamData ) : Boolean
8790:   Function NtfsFindStreamClose( var Data : TFindStreamData ) : Boolean
8791:   Function NtfsCreateHardLink( const LinkFileName, ExistingFileName : string ) : Boolean
8792:   AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end
8793:   Function NtfsGetHardLinkInfo( const FileName : string; var Info : TNtfsHardLinkInfo ) : Boolean
8794:   Function NtfsFindHardLinks(const Path:string;const FileIndexHigh,FileIndexLow:Cardinal;const
8795:   List:TStrings):Bool;
8796:   Function NtfsDeleteHardLinks( const FileName : string ) : Boolean
8797:   Function JclAppInstances : TJclAppInstances;
8798:   Function JclAppInstances1( const UniqueAppIdGuidStr : string ) : TJclAppInstances;
8799:   Procedure ReadMessageCheck( var Message: TMessage;const IgnoredOriginatorWnd: HWND ) : TJclAppInstDataKind
8800:   Procedure ReadMessageData( const Message : TMessage; var Data : __Pointer; var Size : Integer )
8801:   Procedure ReadMessageString( const Message : TMessage; var S : string )
8802:
8803:
8804: (*-----*)
8805: procedure SIRegister_JclGraphics(CL: TPSPascalCompiler);
8806: begin
8807:   FindClass ('TOBJECT'), 'EJclGraphicsError
8808:   TDynDynIntegerArrayArray', 'array of TDynIntegerArray
8809:   TDynPointArray', 'array of TPoint
8810:   TDynDynPointArrayArray', 'array of TDynPointArray
8811:   TPointF', 'record X : Single; Y : Single; end
8812:   TDynPointArrayF', 'array of TPointF
8813:   TDrawMode2', '( dmOpaque, dmBlend )
8814:   TStretchFilter2', '( sfNearest, sfLinear, sfSpline )
8815:   TConversionKind', '( ckRed, ckGreen, ckBlue, ckAlpha, ckUniformRGB, ckWeightedRGB )

```

```

8816: TResamplingFilter', '( rfBox, rfTriangle, rfHermite, rfBell, rfSpline, rfLanczos3, rfMitchell )'
8817: TMatrix3d', 'record array[0..2,0..2] of extended end'
8818: TDynDynPointArrayArrayF', 'array of TDynPointArrayF'
8819: TScanLine', 'array of Integer'
8820: TScanLines', 'array of TScanLine'
8821: TColorChannel', '( ccRed, ccGreen, ccBlue, ccAlpha )'
8822: TGradientDirection', '( gdVertical, gdHorizontal )'
8823: TPolyFillMode', '( fmAlternate, fmWinding )'
8824: TJclRegionCombineOperator', '( coAnd, coDiff, coOr, coXor )'
8825: TJclRegionBitmapMode', '( rmInclude, rmExclude )'
8826: TJclRegionKind', '( rkNull, rkSimple, rkComplex, rkError )'
8827: SIRegister_TJclDesktopCanvas(CL);
8828: FindClass('TOBJECT'), TJclRegion
8829: SIRegister_TJclRegionInfo(CL);
8830: SIRegister_TJclRegion(CL);
8831: SIRegister_TJclThreadPersistent(CL);
8832: SIRegister_TJclCustomMap(CL);
8833: SIRegister_TJclBitmap32(CL);
8834: SIRegister_TJclByteMap(CL);
8835: SIRegister_TJclTransformation(CL);
8836: SIRegister_TJclLinearTransformation(CL);
8837: Procedure Stretch(NewWidth,
NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Source:TGraphic;Target:TBitmap);
8838: Procedure Stretch1(NewWidth,NewHeight:Cardinal;Filter:TResamplingFilter;Radius:Single;Bitmap:TBitmap);
8839: Procedure DrawBitmap( DC : HDC; Bitmap : HBitmap; X, Y, Width, Height : Integer )
8840: Function GetAntialiasedBitmap( const Bitmap : TBitmap ) : TBitmap
8841: Procedure BitmapToJpeg( const FileName : string )
8842: Procedure JpegToBitmap( const FileName : string )
8843: Function ExtractIconCount( const FileName : string ) : Integer
8844: Function BitmapToIconJ( Bitmap : HBITMAP; cx, cy : Integer ) : HICON
8845: Function IconToBitmapJ( Icon : HICON ) : HBITMAP
8846: Procedure
BlockTransfer(Dst:TJclBitmap32;DstX:Int;DstY:Int;Src:TJclBitmap32;SrcRect:TRect;CombineOp:TDrawMode)
8847: Procedure StretchTransfer(Dst:TJclBitmap32;
DstRect:TRect;Src:TJclBitmap32;SrcRect:TRect;StretchFilter:TStretchFilter; CombineOp : TDrawMode)
8848: Procedure Transform( Dst, Src : TJclBitmap32; SrcRect : TRect; Transformation : TJclTransformation )
8849: Procedure SetBorderTransparent( ABitmap : TJclBitmap32; ARect : TRect )
8850: Function FillGradient( DC:HDC; ARect:TRect; ColorCount:Integer; StartColor,EndColor:TColor;ADirection : TGradientDirection ) : Boolean;
8851: Function CreateRegionFromBitmap(Bitmap: TBitmap; RegionColor:TColor;
RegionBitmapMode:TJclRegionBitmapMode) : HRGN
8852: Procedure ScreenShot( bm : TBitmap; Left, Top, Width, Height : Integer; Window : HWND );
8853: Procedure ScreenShot1( bm : TBitmap );
8854: Procedure PolyLineTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32 )
8855: Procedure PolyLineAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32 )
8856: Procedure PolyLineFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32 )
8857: Procedure PolygonTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32 )
8858: Procedure PolygonAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32 )
8859: Procedure PolygonFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32 )
8860: Procedure PolyPolygonTS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8861: Procedure PolyPolygonAS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8862: Procedure PolyPolygonFS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArrayF;Color:TColor32);
8863: Procedure AlphaToGrayscale( Dst, Src : TJclBitmap32 )
8864: Procedure IntensityToAlpha( Dst, Src : TJclBitmap32 )
8865: Procedure Invert( Dst, Src : TJclBitmap32 )
8866: Procedure InvertRGB( Dst, Src : TJclBitmap32 )
8867: Procedure ColorToGrayscale( Dst, Src : TJclBitmap32 )
8868: Procedure ApplyLUT( Dst, Src : TJclBitmap32; const LUT : TLUT8 )
8869: Procedure SetGamma( Gamma : Single )
8870: end;
8871:
8872: (*-----*)
8873: procedure SIRegister_JclSynch(CL: TPSPascalCompiler);
8874: begin
8875: Function LockedAdd( var Target : Integer; Value : Integer ) : Integer
8876: Function LockedCompareExchange( var Target : Integer; Exch, Comp : Integer ) : Integer;
8877: Function LockedCompareExchange1( var Target : __Pointer; Exch, Comp : __Pointer ) : Pointer;
8878: Function LockedDec( var Target : Integer ) : Integer
8879: Function LockedExchange( var Target : Integer; Value : Integer ) : Integer
8880: Function LockedExchangeAdd( var Target : Integer; Value : Integer ) : Integer
8881: Function LockedExchangeDec( var Target : Integer ) : Integer
8882: Function LockedExchangeInc( var Target : Integer ) : Integer
8883: Function LockedExchangeSub( var Target : Integer; Value : Integer ) : Integer
8884: Function LockedInc( var Target : Integer ) : Integer
8885: Function LockedSub( var Target : Integer; Value : Integer ) : Integer
8886: TJclWaitResult', '( wrAbandoned, wrError, wrIoCompletion, wrSignaled, wrTimeout )
8887: SIRegister_TJclDispatcherObject(CL);
8888: Function WaitForMultipleObjects(const Objects:array of
TJclDispatcherObject;WaitAll:Bool;TimeOut:Cardinal):Cardinal;
8889: Function WaitAlertableForMultipleObjects(const Objects : array of TJclDispatcherObject; WaitAll:Bool;
TimeOut : Cardinal):Cardinal
8890: SIRegister_TJclCriticalSection(CL);
8891: SIRegister_TJclCriticalSectionEx(CL);
8892: SIRegister_TJclEvent(CL);
8893: SIRegister_TJclWaitableTimer(CL);
8894: SIRegister_TJclSemaphore(CL);
8895: SIRegister_TJclMutex(CL);
8896: POptexSharedInfo', '^POptexSharedInfo // will not work
8897: TOptexSharedInfo', 'record SpinCount:Int; LockCount: Int; ThreadId:Longword; RecursionCount:Int; end

```

```

8898: SIRegister_TJclOptex(CL);
8899: TMrewPreferred', '( mpReaders, mpWriters, mpEqual )
8900: TMrewThreadInfo', 'record ThreadId : Longword; RecursionCount: Integer; Reader : Boolean; end
8901: TMrewThreadInfoArray', 'array of TMrewThreadInfo
8902: SIRegister_TJclMultiReadExclusiveWrite(CL);
8903: PMetSectSharedInfo', '^TMetSectSharedInfo // will not work
8904: TMetSectSharedInfo', 'record Initialized : LongBool; SpinLock : '
8905: +'Longint; ThreadsWaiting : Longint; AvailableCount : Longint; MaximumCount : Longint; end
8906: PMeteredSection', '^TMeteredSection // will not work
8907: TMeteredSection', 'record Event : THandle; FileMap : THandle; SharedInfo : PMetSectSharedInfo; end
8908: SIRegister_TJclMeteredSection(CL);
8909: TEventInfo', 'record EventType : Longint; Signaled : LongBool; end
8910: TMutexInfo', 'record SignalState : Longint; Owned : Boolean; Abandoned : Boolean; end
8911: TSemaphoreCounts', 'record CurrentCount : Longint; MaximumCount: Longint; end
8912: TTimerInfo', 'record Remaining : TLargeInteger; Signaled : LongBool; end
8913: Function QueryCriticalSection( CS : TJclCriticalSection; var Info : TRTCriticalSection) : Boolean
8914: Function QueryEvent( Handle : THandle; var Info : TEventInfo) : Boolean
8915: Function QueryMutex( Handle : THandle; var Info : TMutexInfo) : Boolean
8916: Function QuerySemaphore( Handle : THandle; var Info : TSemaphoreCounts) : Boolean
8917: Function QueryTimer( Handle : THandle; var Info : TTimerInfo) : Boolean
8918: FindClass('TOBJECT'), 'EJclWin32HandleObjectError
8919: FindClass('TOBJECT'), 'EJclDispatcherObjectError
8920: FindClass('TOBJECT'), 'EJclCriticalSectionError
8921: FindClass('TOBJECT'), 'EJclEventError
8922: FindClass('TOBJECT'), 'EJclWaitableTimerError
8923: FindClass('TOBJECT'), 'EJclSemaphoreError
8924: FindClass('TOBJECT'), 'EJclMutexError
8925: FindClass('TOBJECT'), 'EJclMeteredSectionError
8926: end;
8927:
8928:
8929: //*****unit uPSI_mORMotReport;
8930: Procedure SetCurrentPrinterAsDefault
8931: Function CurrentPrinterName : string
8932: Function mCurrentPrinterPaperSize : string
8933: Procedure UseDefaultPrinter
8934:
8935: procedure SIRegisterTSTREAM(C1: TPSPascalCompiler);
8936: begin
8937:   with FindClass('TOBJECT'), 'TStream') do begin
8938:     IsAbstract := True;
8939:     //RegisterMethod('Function Read( var Buffer, Count : Longint) : Longint
8940:     // RegisterMethod('Function Write( const Buffer, Count : Longint) : Longint
8941:     function Read(Buffer:String;Count:LongInt):LongInt
8942:     function Write(Buffer:String;Count:LongInt):LongInt
8943:     function ReadString(Buffer:String;Count:LongInt):LongInt      //FileStream
8944:     function WriteString(Buffer:String;Count:LongInt):LongInt
8945:     function ReadInt(Buffer:integer;Count:LongInt):LongInt
8946:     function WriteInt(Buffer:integer;Count:LongInt):LongInt
8947:     function ReadByteArray(Buffer:TByteArray;Count:LongInt):LongInt';
8948:     function WriteByteArray(Buffer:TByteArray;Count:LongInt):LongInt';
8949:
8950:     procedure ReadAB(Buffer: TByteArray;Count:LongInt)
8951:     procedure WriteAB(Buffer: TByteArray;Count:LongInt)
8952:     procedure ReadABD(Buffer: TByteDynArray;Count:LongInt)
8953:     procedure WriteABD(Buffer: TByteDynArray;Count:LongInt)
8954:     procedure ReadAC(Buffer: TCharArray;Count:LongInt)
8955:     procedure WriteAC(Buffer: TCharArray;Count:LongInt)
8956:     procedure ReadACD(Buffer: TCharDynArray;Count:LongInt)
8957:     procedure WriteACD(Buffer: TCharDynArray;Count:LongInt)
8958:
8959:     function Seek(Offset:LongInt;Origin:Word):LongInt
8960:     procedure ReadBuffer(Buffer:String;Count:LongInt)
8961:     procedure WriteBuffer(Buffer:String;Count:LongInt)
8962:     procedure ReadBufferInt(Buffer:Integer;Count:LongInt)');
8963:     procedure WriteBufferInt(Buffer:Integer;Count:LongInt)';
8964:     procedure ReadBufferFloat(Buffer:Double;Count:LongInt)';
8965:     procedure WriteBufferFloat(Buffer:Double;Count:LongInt)';
8966:
8967:     procedure ReadBufferAB(Buffer: TByteArray;Count:LongInt)
8968:     procedure WriteBufferAB(Buffer: TByteArray;Count:LongInt)
8969:     procedure ReadBufferABD(Buffer: TByteDynArray;Count:LongInt)
8970:     procedure WriteBufferABD(Buffer: TByteDynArray;Count:LongInt)
8971:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
8972:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
8973:     procedure ReadBufferAWD(Buffer: TWordDynArray;Count:LongInt)
8974:     procedure WriteBufferAWD(Buffer: TWordDynArray;Count:LongInt)
8975:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
8976:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
8977:     procedure ReadBufferAC(Buffer: TCharArray;Count:LongInt)
8978:     procedure WriteBufferAC(Buffer: TCharArray;Count:LongInt)
8979:     procedure ReadBufferACD(Buffer: TCharDynArray;Count:LongInt)
8980:     procedure WriteBufferACD(Buffer: TCharDynArray;Count:LongInt)
8981:
8982:     procedure ReadBufferP(Buffer: PChar;Count:LongInt)
8983:     procedure WriteBufferP(Buffer: PChar;Count:LongInt)
8984:     procedure ReadBufferO(Buffer: TObject;Count:LongInt)';
8985:     procedure WriteBufferO(Buffer: TObject;Count:LongInt)';
8986:   //READBUFFERAC

```

```

8987:   function InstanceSize: Longint
8988:   Procedure FixupResourceHeader( FixupInfo : Integer)
8989:   Procedure ReadResHeader
8990:
8991: {$IFDEF DELPHI4UP}
8992:   function CopyFrom(Source:TStream;Count:Int64):LongInt
8993: {$ELSE}
8994:   function CopyFrom(Source:TStream;Count:Integer):LongInt
8995: {$ENDIF}
8996: RegisterProperty('Position', 'LongInt', iptrw);
8997: RegisterProperty('Size', 'LongInt', iptrw);
8998: end;
8999: end;
9000:
9001: { ****
9002: Unit DMATH - Interface for DMATH.DLL
9003: **** }
9004: // see more docs/dmath_manual.pdf
9005:
9006:
9007: Function InitEval : Integer
9008: Procedure SetVariable( VarName : Char; Value : Float)
9009: Procedure SetFunction( FuncName : String; Wrapper : TWrapper)
9010: Function Eval( ExpressionString : string) : Float
9011:
9012: unit dmath; //types are in built, others are external in DLL
9013: interface
9014: {$IFDEF DELPHI}
9015: uses
9016:   StdCtrls, Graphics;
9017: {$ENDIF}
9018: { -----
9019:   Types and constants
9020: ----- }
9021: {$i types.inc}
9022: { -----
9023:   Error handling
9024: ----- }
9025: procedure SetErrCode(ErrCode : Integer); external 'dmath';
9026: { Sets the error code }
9027: function DefaultVal(ErrCode : Integer; DefVal : Float) : Float; external 'dmath';
9028: { Sets error code and default function value }
9029: function MathErr : Integer; external 'dmath';
9030: { Returns the error code }
9031: { -----
9032:   Dynamic arrays
9033: ----- }
9034: procedure SetAutoInit(AutoInit : Boolean); external 'dmath';
9035: { Sets the auto-initialization of arrays }
9036: procedure DimVector(var V : TVector; Ub : Integer); external 'dmath';
9037: { Creates floating point vector V[0..Ub] }
9038: procedure DimIntVector(var V : TIntVector; Ub : Integer); external 'dmath';
9039: { Creates integer vector V[0..Ub] }
9040: procedure DimCompVector(var V : TCompVector; Ub : Integer); external 'dmath';
9041: { Creates complex vector V[0..Ub] }
9042: procedure DimBoolVector(var V : TBoolVector; Ub : Integer); external 'dmath';
9043: { Creates boolean vector V[0..Ub] }
9044: procedure DimStrVector(var V : TStringVector; Ub : Integer); external 'dmath';
9045: { Creates string vector V[0..Ub] }
9046: procedure DimMatrix(var A : TMatrix; Ub1, Ub2 : Integer); external 'dmath';
9047: { Creates floating point matrix A[0..Ub1, 0..Ub2] }
9048: procedure DimIntMatrix(var A : TIntMatrix; Ub1, Ub2 : Integer); external 'dmath';
9049: { Creates integer matrix A[0..Ub1, 0..Ub2] }
9050: procedure DimCompMatrix(var A : TCompMatrix; Ub1, Ub2 : Integer); external 'dmath';
9051: { Creates complex matrix A[0..Ub1, 0..Ub2] }
9052: procedure DimBoolMatrix(var A : TBoolMatrix; Ub1, Ub2 : Integer); external 'dmath';
9053: { Creates boolean matrix A[0..Ub1, 0..Ub2] }
9054: procedure DimStrMatrix(var A : TStringMatrix; Ub1, Ub2 : Integer); external 'dmath';
9055: { Creates string matrix A[0..Ub1, 0..Ub2] }
9056: { -----
9057:   Minimum, maximum, sign and exchange
9058: ----- }
9059: function FMin(X, Y : Float) : Float; external 'dmath';
9060: { Minimum of 2 reals }
9061: function FMax(X, Y : Float) : Float; external 'dmath';
9062: { Maximum of 2 reals }
9063: function IMin(X, Y : Integer) : Integer; external 'dmath';
9064: { Minimum of 2 integers }
9065: function IMax(X, Y : Integer) : Integer; external 'dmath';
9066: { Maximum of 2 integers }
9067: function Sgn(X : Float) : Integer; external 'dmath';
9068: { Sign (returns 1 if X = 0) }
9069: function Sgn0(X : Float) : Integer; external 'dmath';
9070: { Sign (returns 0 if X = 0) }
9071: function DSgn(A, B : Float) : Float; external 'dmath';
9072: { Sgn(B) * |A| }
9073: procedure FSwap(var X, Y : Float); external 'dmath';
9074: { Exchange 2 reals }
9075: procedure ISwap(var X, Y : Integer); external 'dmath';

```

```

9076: { Exchange 2 integers }
9077: { -----
9078:   Rounding functions
9079:   -----
9080:   function RoundN(X : Float; N : Integer) : Float; external 'dmath';
9081:   { Rounds X to N decimal places }
9082:   function Ceil(X : Float) : Integer; external 'dmath';
9083:   { Ceiling function }
9084:   function Floor(X : Float) : Integer; external 'dmath';
9085:   { Floor function }
9086:   -----
9087:   Logarithms, exponentials and power
9088:   -----
9089:   function Exp(X : Float) : Float; external 'dmath';
9090:   { Exponential }
9091:   function Exp2(X : Float) : Float; external 'dmath';
9092:   { 2^X }
9093:   function Exp10(X : Float) : Float; external 'dmath';
9094:   { 10^X }
9095:   function Log(X : Float) : Float; external 'dmath';
9096:   { Natural log }
9097:   function Log2(X : Float) : Float; external 'dmath';
9098:   { Log, base 2 }
9099:   function Log10(X : Float) : Float; external 'dmath';
9100:   { Decimal log }
9101:   function LogA(X, A : Float) : Float; external 'dmath';
9102:   { Log, base A }
9103:   function IntPower(X : Float; N : Integer) : Float; external 'dmath';
9104:   { X^N }
9105:   function Power(X, Y : Float) : Float; external 'dmath';
9106:   { X^Y, X >= 0 }
9107:   -----
9108:   Trigonometric functions
9109:   -----
9110:   function Pythag(X, Y : Float) : Float; external 'dmath';
9111:   { Sqrt(X^2 + Y^2) }
9112:   function FixAngle(Theta : Float) : Float; external 'dmath';
9113:   { Set Theta in -Pi..Pi }
9114:   function Tan(X : Float) : Float; external 'dmath';
9115:   { Tangent }
9116:   function ArcSin(X : Float) : Float; external 'dmath';
9117:   { Arc sinus }
9118:   function ArcCos(X : Float) : Float; external 'dmath';
9119:   { Arc cosinus }
9120:   function ArcTan2(Y, X : Float) : Float; external 'dmath';
9121:   { Angle (Ox, OM) with M(X,Y) }
9122:   -----
9123:   Hyperbolic functions
9124:   -----
9125:   function Sinh(X : Float) : Float; external 'dmath';
9126:   { Hyperbolic sine }
9127:   function Cosh(X : Float) : Float; external 'dmath';
9128:   { Hyperbolic cosine }
9129:   function Tanh(X : Float) : Float; external 'dmath';
9130:   { Hyperbolic tangent }
9131:   function ArcSinh(X : Float) : Float; external 'dmath';
9132:   { Inverse hyperbolic sine }
9133:   function ArcCosh(X : Float) : Float; external 'dmath';
9134:   { Inverse hyperbolic cosine }
9135:   function ArcTanh(X : Float) : Float; external 'dmath';
9136:   { Inverse hyperbolic tangent }
9137:   procedure SinhCosh(X : Float; var SinhX, CoshX : Float); external 'dmath';
9138:   { Sinh & Cosh }
9139:   -----
9140:   Gamma function and related functions
9141:   -----
9142:   function Fact(N : Integer) : Float; external 'dmath';
9143:   { Factorial }
9144:   function SgnGamma(X : Float) : Integer; external 'dmath';
9145:   { Sign of Gamma function }
9146:   function Gamma(X : Float) : Float; external 'dmath';
9147:   { Gamma function }
9148:   function LnGamma(X : Float) : Float; external 'dmath';
9149:   { Logarithm of Gamma function }
9150:   function Stirling(X : Float) : Float; external 'dmath';
9151:   { Stirling's formula for the Gamma function }
9152:   function StirLog(X : Float) : Float; external 'dmath';
9153:   { Approximate Ln(Gamma) by Stirling's formula, for X >= 13 }
9154:   function DiGamma(X : Float) : Float; external 'dmath';
9155:   { Digamma function }
9156:   function TriGamma(X : Float) : Float; external 'dmath';
9157:   { Trigamma function }
9158:   function IGamma(A, X : Float) : Float; external 'dmath';
9159:   { Incomplete Gamma function }
9160:   function JGamma(A, X : Float) : Float; external 'dmath';
9161:   { Complement of incomplete Gamma function }
9162:   function InvGamma(A, P : Float) : Float; external 'dmath';
9163:   { Inverse of incomplete Gamma function }
9164:   function Erf(X : Float) : Float; external 'dmath';

```

```

9165: { Error function }
9166: function Erfc(X : Float) : Float; external 'dmath';
9167: { Complement of error function }
9168: {
9169:   Beta function and related functions
9170:   -----
9171:   function Beta(X, Y : Float) : Float; external 'dmath';
9172:   { Beta function }
9173:   function IBeta(A, B, X : Float) : Float; external 'dmath';
9174:   { Incomplete Beta function }
9175:   function InvBeta(A, B, Y : Float) : Float; external 'dmath';
9176:   { Inverse of incomplete Beta function }
9177:   {
9178:     Lambert's function
9179:     -----
9180:     function LambertW(X : Float; UBranch, Offset : Boolean) : Float; external 'dmath';
9181:   -----
9182:   Binomial distribution
9183:   -----
9184:   function Binomial(N, K : Integer) : Float; external 'dmath';
9185:   { Binomial coefficient C(N,K) }
9186:   function PBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9187:   { Probability of binomial distribution }
9188:   function FBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9189:   { Cumulative probability for binomial distrib. }
9190:   {
9191:     Poisson distribution
9192:     -----
9193:     function PPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9194:     { Probability of Poisson distribution }
9195:     function FPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9196:     { Cumulative probability for Poisson distrib. }
9197:     {
9198:       Exponential distribution
9199:       -----
9200:       function DExpo(A, X : Float) : Float; external 'dmath';
9201:       { Density of exponential distribution with parameter A }
9202:       function FExpo(A, X : Float) : Float; external 'dmath';
9203:       { Cumulative probability function for exponential dist. with parameter A }
9204:       {
9205:         Standard normal distribution
9206:         -----
9207:         function DNorm(X : Float) : Float; external 'dmath';
9208:         { Density of standard normal distribution }
9209:         function FNorm(X : Float) : Float; external 'dmath';
9210:         { Cumulative probability for standard normal distrib. }
9211:         function PNorm(X : Float) : Float; external 'dmath';
9212:         { Prob(|U| > X) for standard normal distrib. }
9213:         function InvNorm(P : Float) : Float; external 'dmath';
9214:         { Inverse of standard normal distribution }
9215:         {
9216:           Student's distribution
9217:           -----
9218:           function DStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9219:           { Density of Student distribution with Nu d.o.f. }
9220:           function FStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9221:           { Cumulative probability for Student distrib. with Nu d.o.f. }
9222:           function PStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9223:           { Prob(|t| > X) for Student distrib. with Nu d.o.f. }
9224:           function InvStudent(Nu : Integer; P : Float) : Float; external 'dmath';
9225:           { Inverse of Student's t-distribution function }
9226:           {
9227:             Khi-2 distribution
9228:             -----
9229:             function DKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9230:             { Density of Khi-2 distribution with Nu d.o.f. }
9231:             function FKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9232:             { Cumulative prob. for Khi-2 distrib. with Nu d.o.f. }
9233:             function PKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9234:             { Prob(Khi2 > X) for Khi-2 distrib. with Nu d.o.f. }
9235:             function InvKhi2(Nu : Integer; P : Float) : Float; external 'dmath';
9236:             { Inverse of Khi-2 distribution function }
9237:             {
9238:               Fisher-Snedecor distribution
9239:               -----
9240:               function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9241:               { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
9242:               function FSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9243:               { Cumulative prob. for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9244:               function PSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9245:               { Prob(F > X) for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9246:               function InvSnedecor(Nu1, Nu2 : Integer; P : Float) : Float; external 'dmath';
9247:               { Inverse of Snedecor's F-distribution function }
9248:               {
9249:                 Beta distribution
9250:                 -----
9251:                 function DBeta(A, B, X : Float) : Float; external 'dmath';
9252:                 { Density of Beta distribution with parameters A and B }
9253:                 function FBeta(A, B, X : Float) : Float; external 'dmath';

```

```

9254: { Cumulative probability for Beta distrib. with param. A and B }
9255: { -----
9256:   Gamma distribution
9257: ----- }
9258: function DGamma(A, B, X : Float) : Float; external 'dmath';
9259: { Density of Gamma distribution with parameters A and B }
9260: function FGamma(A, B, X : Float) : Float; external 'dmath';
9261: { Cumulative probability for Gamma distrib. with param. A and B }
9262: { -----
9263:   Expression evaluation
9264: ----- }
9265: function InitEval : Integer; external 'dmath';
9266: { Initializes built-in functions and returns their number }
9267: function Eval(ExpressionString : String) : Float; external 'dmath';
9268: { Evaluates an expression at run-time }
9269: procedure SetVariable(VarName : Char; Value : Float); external 'dmath';
9270: { Assigns a value to a variable }
9271: procedure SetFunction(FuncName : String; Wrapper : TWrapper); external 'dmath';
9272: { Adds a function to the parser }
9273: { -----
9274:   Matrices and linear equations
9275: ----- }
9276: procedure GaussJordan(A          : TMatrix;
9277:                         Lb, Ubl, Ub2 : Integer;
9278:                         var Det      : Float); external 'dmath';
9279: { Transforms a matrix according to the Gauss-Jordan method }
9280: procedure LinEq(A          : TMatrix;
9281:                     B          : TVector;
9282:                     Lb, Ub   : Integer;
9283:                     var Det    : Float); external 'dmath';
9284: { Solves a linear system according to the Gauss-Jordan method }
9285: procedure Cholesky(A, L : TMatrix; Lb, Ub : Integer); external 'dmath';
9286: { Cholesky factorization of a positive definite symmetric matrix }
9287: procedure LU_Decom(A : TMatrix; Lb, Ub : Integer); external 'dmath';
9288: { LU decomposition }
9289: procedure LU_Solve(A       : TMatrix;
9290:                       B       : TVector;
9291:                       Lb, Ub : Integer;
9292:                       X       : TVector); external 'dmath';
9293: { Solution of linear system from LU decomposition }
9294: procedure QR_Decom(A       : TMatrix;
9295:                       Lb, Ubl, Ub2 : Integer;
9296:                       R       : TMatrix); external 'dmath';
9297: { QR decomposition }
9298: procedure QR_Solve(Q, R     : TMatrix;
9299:                       B       : TVector;
9300:                       Lb, Ubl, Ub2 : Integer;
9301:                       X       : TVector); external 'dmath';
9302: { Solution of linear system from QR decomposition }
9303: procedure SV_Decom(A       : TMatrix;
9304:                       Lb, Ubl, Ub2 : Integer;
9305:                       S       : TVector;
9306:                       V       : TMatrix); external 'dmath';
9307: { Singular value decomposition }
9308: procedure SV_SetZero(S    : TVector;
9309:                         Lb, Ub : Integer;
9310:                         Tol    : Float); external 'dmath';
9311: { Set lowest singular values to zero }
9312: procedure SV_Solve(U     : TMatrix;
9313:                       S     : TVector;
9314:                       V     : TMatrix;
9315:                       B     : TVector;
9316:                       Lb, Ubl, Ub2 : Integer;
9317:                       X     : TVector); external 'dmath';
9318: { Solution of linear system from SVD }
9319: procedure SV_Approx(U    : TMatrix;
9320:                       S    : TVector;
9321:                       V    : TMatrix;
9322:                       Lb, Ubl, Ub2 : Integer;
9323:                       A    : TMatrix); external 'dmath';
9324: { Matrix approximation from SVD }
9325: procedure EigenVals(A   : TMatrix;
9326:                         Lb, Ub : Integer;
9327:                         Lambda : TCompVector); external 'dmath';
9328: { Eigenvalues of a general square matrix }
9329: procedure EigenVect(A   : TMatrix;
9330:                         Lb, Ub : Integer;
9331:                         Lambda : TCompVector;
9332:                         V     : TMatrix); external 'dmath';
9333: { Eigenvalues and eigenvectors of a general square matrix }
9334: procedure EigenSym(A   : TMatrix;
9335:                         Lb, Ub : Integer;
9336:                         Lambda : TVector;
9337:                         V     : TMatrix); external 'dmath';
9338: { Eigenvalues and eigenvectors of a symmetric matrix (SVD method) }
9339: procedure Jacobi(A    : TMatrix;
9340:                      Lb, Ub, MaxIter : Integer;
9341:                      Tol    : Float;
9342:                      Lambda : TVector);

```

```

9343:           V           : TMatrix); external 'dmath';
9344: { Eigenvalues and eigenvectors of a symmetric matrix (Jacobi method) }
9345: { -----
9346:   Optimization
9347: -----
9348: procedure MinBrack(Func          : TFunc;
9349:                       var A, B, C, Fa, Fb, Fc : Float); external 'dmath';
9350: { Brackets a minimum of a function }
9351: procedure GoldSearch(Func        : TFunc;
9352:                         A, B       : Float;
9353:                         MaxIter    : Integer;
9354:                         Tol        : Float;
9355:                         var Xmin, Ymin : Float); external 'dmath';
9356: { Minimization of a function of one variable (golden search) }
9357: procedure LinMin(Func         : TFuncNVar;
9358:                      X, DeltaX : TVector;
9359:                      Lb, Ub    : Integer;
9360:                      var R     : Float;
9361:                      MaxIter   : Integer;
9362:                      Tol       : Float;
9363:                      var F_min : Float); external 'dmath';
9364: { Minimization of a function of several variables along a line }
9365: procedure Newton(Func        : TFuncNVar;
9366:                      HessGrad : THessGrad;
9367:                      X        : TVector;
9368:                      Lb, Ub   : Integer;
9369:                      MaxIter  : Integer;
9370:                      Tol      : Float;
9371:                      var F_min : Float;
9372:                      G        : TVector;
9373:                      H_inv    : TMatrix;
9374:                      var Det   : Float); external 'dmath';
9375: { Minimization of a function of several variables (Newton's method) }
9376: procedure SaveNewton(FileName : string); external 'dmath';
9377: { Save Newton iterations in a file }
9378: procedure Marquardt(Func      : TFuncNVar;
9379:                         HessGrad : THessGrad;
9380:                         X        : TVector;
9381:                         Lb, Ub   : Integer;
9382:                         MaxIter  : Integer;
9383:                         Tol      : Float;
9384:                         var F_min : Float;
9385:                         G        : TVector;
9386:                         H_inv    : TMatrix;
9387:                         var Det   : Float); external 'dmath';
9388: { Minimization of a function of several variables (Marquardt's method) }
9389: procedure SaveMarquardt(FileName : string); external 'dmath';
9390: { Save Marquardt iterations in a file }
9391: procedure BFGS(Func       : TFuncNVar;
9392:                     Gradient : TGradient;
9393:                     X        : TVector;
9394:                     Lb, Ub   : Integer;
9395:                     MaxIter  : Integer;
9396:                     Tol      : Float;
9397:                     var F_min : Float;
9398:                     G        : TVector;
9399:                     H_inv    : TMatrix); external 'dmath';
9400: { Minimization of a function of several variables (BFGS method) }
9401: procedure SaveBFGS(FileName : string); external 'dmath';
9402: { Save BFGS iterations in a file }
9403: procedure Simplex(Func     : TFuncNVar;
9404:                         X        : TVector;
9405:                         Lb, Ub   : Integer;
9406:                         MaxIter  : Integer;
9407:                         Tol      : Float;
9408:                         var F_min : Float); external 'dmath';
9409: { Minimization of a function of several variables (Simplex) }
9410: procedure SaveSimplex(FileName : string); external 'dmath';
9411: { Save Simplex iterations in a file }
9412: { -----
9413:   Nonlinear equations
9414: -----
9415: procedure RootBrack(Func      : TFunc;
9416:                         var X, Y, FX, FY : Float); external 'dmath';
9417: { Brackets a root of function Func between X and Y }
9418: procedure Bisect(Func      : TFunc;
9419:                      var X, Y : Float;
9420:                      MaxIter : Integer;
9421:                      Tol     : Float;
9422:                      var F   : Float); external 'dmath';
9423: { Bisection method }
9424: procedure Secant(Func     : TFunc;
9425:                      var X, Y : Float;
9426:                      MaxIter : Integer;
9427:                      Tol     : Float;
9428:                      var F   : Float); external 'dmath';
9429: { Secant method }
9430: procedure NewtEq(Func, Deriv : TFunc;
9431:                      var X      : Float;

```

```

9432:           MaxIter      : Integer;
9433:           Tol          : Float;
9434:           var F        : Float); external 'dmath';
9435: { Newton-Raphson method for a single nonlinear equation }
9436: procedure NewtEqs(Equations : TEquations;
9437:                         Jacobian   : TJacobian;
9438:                         X, F       : TVector;
9439:                         Lb, Ub     : Integer;
9440:                         MaxIter    : Integer;
9441:                         Tol        : Float); external 'dmath';
9442: { Newton-Raphson method for a system of nonlinear equations }
9443: procedure Broyden(Equations : TEquations;
9444:                         X, F       : TVector;
9445:                         Lb, Ub     : Integer;
9446:                         MaxIter    : Integer;
9447:                         Tol        : Float); external 'dmath';
9448: { Broyden's method for a system of nonlinear equations }
9449: { -----
9450:   Polynomials and rational fractions
9451: ----- }
9452: function Poly(X      : Float;
9453:                   Coef    : TVector;
9454:                   Deg    : Integer) : Float; external 'dmath';
9455: { Evaluates a polynomial }
9456: function RFrac(X      : Float;
9457:                   Coef    : TVector;
9458:                   Degl, Deg2 : Integer) : Float; external 'dmath';
9459: { Evaluates a rational fraction }
9460: function RootPoli(A, B : Float;
9461:                         var X : Float) : Integer; external 'dmath';
9462: { Solves the linear equation A + B * X = 0 }
9463: function RootPol2(Coef : TVector;
9464:                         Z      : TCompVector) : Integer; external 'dmath';
9465: { Solves a quadratic equation }
9466: function RootPol3(Coef : TVector;
9467:                         Z      : TCompVector) : Integer; external 'dmath';
9468: { Solves a cubic equation }
9469: function RootPol4(Coef : TVector;
9470:                         Z      : TCompVector) : Integer; external 'dmath';
9471: { Solves a quartic equation }
9472: function RootPol(Coef : TVector;
9473:                         Deg   : Integer;
9474:                         Z      : TCompVector) : Integer; external 'dmath';
9475: { Solves a polynomial equation }
9476: function SetRealRoots(Deg : Integer;
9477:                           Z      : TCompVector;
9478:                           Tol   : Float) : Integer; external 'dmath';
9479: { Set the imaginary part of a root to zero }
9480: procedure SortRoots(Deg : Integer;
9481:                         Z      : TCompVector); external 'dmath';
9482: { Sorts the roots of a polynomial }
9483: { -----
9484:   Numerical integration and differential equations
9485: ----- }
9486: function TrapInt(X, Y : TVector; N : Integer) : Float; external 'dmath';
9487: { Integration by trapezoidal rule }
9488: function GausLeg(Func : TFunc; A, B : Float) : Float; external 'dmath';
9489: { Integral from A to B }
9490: function GausLeg0(Func : TFunc; B : Float) : Float; external 'dmath';
9491: { Integral from 0 to B }
9492: function Convol(Func1, Func2 : TFunc; T : Float) : Float; external 'dmath';
9493: { Convolution product at time T }
9494: procedure ConvTrap(Func1, Func2:TFunc; T,Y:TVector; N:Integer);external 'dmath';
9495: { Convolution by trapezoidal rule }
9496: procedure RKF45(F      : TDiffEqs;
9497:                     Neqn      : Integer;
9498:                     Y, Yp     : TVector;
9499:                     var T     : Float;
9500:                     Tout, RelErr, AbsErr : Float;
9501:                     var Flag   : Integer); external 'dmath';
9502: { Integration of a system of differential equations }
9503: { -----
9504:   Fast Fourier Transform
9505: ----- }
9506: procedure FFT(NumSamples      : Integer;
9507:                  InArray, OutArray : TCompVector); external 'dmath';
9508: { Fast Fourier Transform }
9509: procedure IFFT(NumSamples      : Integer;
9510:                  InArray, OutArray : TCompVector); external 'dmath';
9511: { Inverse Fast Fourier Transform }
9512: procedure FFT_Integer(NumSamples      : Integer;
9513:                           RealIn, ImagIn : TIntVector;
9514:                           OutArray     : TCompVector); external 'dmath';
9515: { Fast Fourier Transform for integer data }
9516: procedure FFT_Integer_Cleanup; external 'dmath';
9517: { Clear memory after a call to FFT_Integer }
9518: procedure CalcFrequency(NumSamples,
9519:                           FrequencyIndex : Integer;
9520:                           InArray       : TCompVector;

```

```

9521:           var FFT      : Complex); external 'dmath';
9522: { Direct computation of Fourier transform }
9523: { -----
9524:   Random numbers
9525: ----- }
9526: procedure SetRNG(RNG : RNG_Type); external 'dmath';
9527: { Select generator }
9528: procedure InitGen(Seed : RNG_IntType); external 'dmath';
9529: { Initialize generator }
9530: function IRanGen : RNG_IntType; external 'dmath';
9531: { 32-bit random integer in [-2^31 .. 2^31 - 1] }
9532: function IRanGen31 : RNG_IntType; external 'dmath';
9533: { 31-bit random integer in [0 .. 2^31 - 1] }
9534: function RanGen1 : Float; external 'dmath';
9535: { 32-bit random real in [0,1] }
9536: function RanGen2 : Float; external 'dmath';
9537: { 32-bit random real in [0,1) }
9538: function RanGen3 : Float; external 'dmath';
9539: { 32-bit random real in (0,1) }
9540: function RanGen53 : Float; external 'dmath';
9541: { 53-bit random real in [0,1) }
9542: procedure InitMWC(Seed : RNG_IntType); external 'dmath';
9543: { Initializes the 'Multiply with carry' random number generator }
9544: function IRanMWC : RNG_IntType; external 'dmath';
9545: { Returns a 32 bit random number in [-2^31 ; 2^31-1] }
9546: procedure InitMT(Seed : RNG_IntType); external 'dmath';
9547: { Initializes Mersenne Twister generator with a seed }
9548: procedure InitMTbyArray(InitKey : array of RNG_LongType;
9549:                           KeyLength : Word); external 'dmath';
9550: { Initialize MT generator with an array InitKey[0..(KeyLength - 1)] }
9551: function IRanMT : RNG_IntType; external 'dmath';
9552: { Random integer from MT generator }
9553: procedure InitUVAGByString(KeyPhrase : string); external 'dmath';
9554: { Initializes the UVAG generator with a string }
9555: procedure InitUVAG(Seed : RNG_IntType); external 'dmath';
9556: { Initializes the UVAG generator with an integer }
9557: function IRanUVAG : RNG_IntType; external 'dmath';
9558: { Random integer from UVAG generator }
9559: function RanGaussStd : Float; external 'dmath';
9560: { Random number from standard normal distribution }
9561: function RanGauss(Mu, Sigma : Float) : Float; external 'dmath';
9562: { Random number from normal distrib. with mean Mu and S. D. Sigma }
9563: procedure RanMult(M       : TVector; L       : TMatrix;
9564:                      Lb, Ub : Integer;
9565:                      X       : TVector); external 'dmath';
9566: { Random vector from multinormal distribution (correlated) }
9567: procedure RanMultIndep(M, S   : TVector;
9568:                          Lb, Ub : Integer;
9569:                          X       : TVector); external 'dmath';
9570: { Random vector from multinormal distribution (uncorrelated) }
9571: procedure InitMHParams(NCycles, MaxSim, SavedSim : Integer); external 'dmath';
9572: { Initializes Metropolis-Hastings parameters }
9573: procedure GetMHParams(var NCycles, MaxSim, SavedSim:Integer); external 'dmath';
9574: { Returns Metropolis-Hastings parameters }
9575: procedure Hastings(Func      : TFuncNVar;
9576:                        T        : Float;
9577:                        X        : TVector;
9578:                        V        : TMatrix;
9579:                        Lb, Ub  : Integer;
9580:                        Xmat    : TMatrix;
9581:                        X_min   : TVector;
9582:                        var F_min : Float); external 'dmath';
9583: { Simulation of a probability density function by Metropolis-Hastings }
9584: procedure InitsSAParams(NT, NS, NCycles : Integer; RT : Float); external 'dmath';
9585: { Initializes Simulated Annealing parameters }
9586: procedure SA_CreateLogFile(FileName : String); external 'dmath';
9587: { Initializes log file }
9588: procedure SimAnn(Func      : TFuncNVar;
9589:                       X, Xmin, Xmax : TVector;
9590:                       Lb, Ub  : Integer;
9591:                       var F_min : Float); external 'dmath';
9592: { Minimization of a function of several var. by simulated annealing }
9593: procedure InitGAParams(NP, NG : Integer; SR, MR, HR : Float); external 'dmath';
9594: { Initializes Genetic Algorithm parameters }
9595: procedure GA_CreateLogFile(FileName : String); external 'dmath';
9596: { Initializes log file }
9597: procedure GenAlg(Func      : TFuncNVar;
9598:                       X, Xmin, Xmax : TVector;
9599:                       Lb, Ub  : Integer;
9600:                       var F_min : Float); external 'dmath';
9601: { Minimization of a function of several var. by genetic algorithm }
9602: { -----
9603:   Statistics
9604:   ----- }
9605: function Mean(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9606: { Mean of sample X }
9607: function Min(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9608: { Minimum of sample X }
9609: function Max(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';

```

```

9610: { Maximum of sample X }
9611: function Median(X : TVector; Lb, Ub : Integer; Sorted : Boolean) : Float; external 'dmath';
9612: { Median of sample X }
9613: function StDev(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9614: { Standard deviation estimated from sample X }
9615: function StDevP(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9616: { Standard deviation of population }
9617: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9618: { Correlation coefficient }
9619: function Skewness(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9620: { Skewness of sample X }
9621: function Kurtosis(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9622: { Kurtosis of sample X }
9623: procedure QSORT(X : TVector; Lb, Ub : Integer); external 'dmath';
9624: { Quick sort (ascending order) }
9625: procedure DQSORT(X : TVector; Lb, Ub : Integer); external 'dmath';
9626: { Quick sort (descending order) }
9627: procedure Interval(X1, X2 : Float;
9628:                      MinDiv, MaxDiv : Integer;
9629:                      var Min, Max, Step : Float); external 'dmath';
9630: { Determines an interval for a set of values }
9631: procedure AutoScale(X : TVector; Lb, Ub : Integer; Scale : TScale;
9632:                       var XMin, XMax, XStep : Float); external 'dmath';
9633: { Finds an appropriate scale for plotting the data in X[Lb..Ub] }
9634: procedure StudIndep(N1, N2 : Integer;
9635:                       M1, M2, S1, S2 : Float;
9636:                       var T : Float;
9637:                       var DoF : Integer); external 'dmath';
9638: { Student t-test for independent samples }
9639: procedure StudPaired(X, Y : TVector;
9640:                        Lb, Ub : Integer;
9641:                        var T : Float;
9642:                        var DoF : Integer); external 'dmath';
9643: { Student t-test for paired samples }
9644: procedure AnOVal(Ns : Integer;
9645:                      N : TIntVector;
9646:                      M, S : TVector;
9647:                      var V_f, V_r, F : Float;
9648:                      var DoF_f, DoF_r : Integer); external 'dmath';
9649: { One-way analysis of variance }
9650: procedure AnOva2(NA, NB, Nobs : Integer;
9651:                      M, S : TMatrix;
9652:                      V, F : TVector;
9653:                      DoF : TIntVector); external 'dmath';
9654: { Two-way analysis of variance }
9655: procedure Snedecor(N1, N2 : Integer;
9656:                       S1, S2 : Float;
9657:                       var F : Float;
9658:                       var DoF1, DoF2 : Integer); external 'dmath';
9659: { Snedecor's F-test (comparison of two variances) }
9660: procedure Bartlett(Ns : Integer;
9661:                       N : TIntVector;
9662:                       S : TVector;
9663:                       var Khi2 : Float;
9664:                       var DoF : Integer); external 'dmath';
9665: { Bartlett's test (comparison of several variances) }
9666: procedure Mann_Whitney(N1, N2 : Integer;
9667:                           X1, X2 : TVector;
9668:                           var U, Eps : Float); external 'dmath';
9669: { Mann-Whitney test }
9670: procedure Wilcoxon(X, Y : TVector;
9671:                       Lb, Ub : Integer;
9672:                       var Ndiff : Integer;
9673:                       var T, Eps : Float); external 'dmath';
9674: { Wilcoxon test }
9675: procedure Kruskal_Wallis(Ns : Integer;
9676:                           N : TIntVector;
9677:                           X : TMatrix;
9678:                           var H : Float;
9679:                           var DoF : Integer); external 'dmath';
9680: { Kruskal-Wallis test }
9681: procedure Khi2_Conform(N_cls : Integer;
9682:                           N_estim : Integer;
9683:                           Obs : TIntVector;
9684:                           Calc : TVector;
9685:                           var Khi2 : Float;
9686:                           var DoF : Integer); external 'dmath';
9687: { Khi-2 test for conformity }
9688: procedure Khi2_Indep(N_lin : Integer;
9689:                           N_col : Integer;
9690:                           Obs : TIntMatrix;
9691:                           var Khi2 : Float;
9692:                           var DoF : Integer); external 'dmath';
9693: { Khi-2 test for independence }
9694: procedure Woolf_Conform(N_cls : Integer;
9695:                           N_estim : Integer;
9696:                           Obs : TIntVector;
9697:                           Calc : TVector;
9698:                           var G : Float;

```

```

9699:           var DoF : Integer); external 'dmath';
9700: { Woolf's test for conformity }
9701: procedure Woolf_Indep(N_lin : Integer;
9702:                           N_col : Integer;
9703:                           Obs : TIntMatrix;
9704:                           var G : Float;
9705:                           var DoF : Integer); external 'dmath';
9706: { Woolf's test for independence }
9707: procedure DimStatClassVector(var C : TStatClassVector;
9708:                                 Ub : Integer); external 'dmath';
9709: { Allocates an array of statistical classes: C[0..Ub] }
9710: procedure Distrib(X : TVector;
9711:                      Lb, Ub : Integer;
9712:                      A, B, H : Float;
9713:                      C : TStatClassVector); external 'dmath';
9714: { Distributes an array X[Lb..Ub] into statistical classes }
9715: { -----
9716:   Linear / polynomial regression
9717:   -----
9718: procedure LinFit(X, Y : TVector;
9719:                     Lb, Ub : Integer;
9720:                     B : TVector;
9721:                     V : TMMatrix); external 'dmath';
9722: { Linear regression :  $Y = B(0) + B(1) * X$  }
9723: procedure WLInFit(X, Y, S : TVector;
9724:                      Lb, Ub : Integer;
9725:                      B : TVector;
9726:                      V : TMMatrix); external 'dmath';
9727: { Weighted linear regression :  $Y = B(0) + B(1) * X$  }
9728: procedure SVDFLinFit(X, Y : TVector;
9729:                        Lb, Ub : Integer;
9730:                        SVDTol : Float;
9731:                        B : TVector;
9732:                        V : TMMatrix); external 'dmath';
9733: { Unweighted linear regression by singular value decomposition }
9734: procedure WSVDLinFit(X, Y, S : TVector;
9735:                        Lb, Ub : Integer;
9736:                        SVDTol : Float;
9737:                        B : TVector;
9738:                        V : TMMatrix); external 'dmath';
9739: { Weighted linear regression by singular value decomposition }
9740: procedure MulFit(X : TMMatrix;
9741:                     Y : TVector;
9742:                     Lb, Ub, Nvar : Integer;
9743:                     ConsTerm : Boolean;
9744:                     B : TVector;
9745:                     V : TMMatrix); external 'dmath';
9746: { Multiple linear regression by Gauss-Jordan method }
9747: procedure WMulFit(X : TMMatrix;
9748:                     Y, S : TVector;
9749:                     Lb, Ub, Nvar : Integer;
9750:                     ConsTerm : Boolean;
9751:                     B : TVector;
9752:                     V : TMMatrix); external 'dmath';
9753: { Weighted multiple linear regression by Gauss-Jordan method }
9754: procedure SVDFit(X : TMMatrix;
9755:                     Y : TVector;
9756:                     Lb, Ub, Nvar : Integer;
9757:                     ConsTerm : Boolean;
9758:                     SVDTol : Float;
9759:                     B : TVector;
9760:                     V : TMMatrix); external 'dmath';
9761: { Multiple linear regression by singular value decomposition }
9762: procedure WSVDFit(X : TMMatrix;
9763:                     Y, S : TVector;
9764:                     Lb, Ub, Nvar : Integer;
9765:                     ConsTerm : Boolean;
9766:                     SVDTol : Float;
9767:                     B : TVector;
9768:                     V : TMMatrix); external 'dmath';
9769: { Weighted multiple linear regression by singular value decomposition }
9770: procedure PolFit(X, Y : TVector;
9771:                     Lb, Ub, Deg : Integer;
9772:                     B : TVector;
9773:                     V : TMMatrix); external 'dmath';
9774: { Polynomial regression by Gauss-Jordan method }
9775: procedure WPolFit(X, Y, S : TVector;
9776:                     Lb, Ub, Deg : Integer;
9777:                     B : TVector;
9778:                     V : TMMatrix); external 'dmath';
9779: { Weighted polynomial regression by Gauss-Jordan method }
9780: procedure SVDPolFit(X, Y : TVector;
9781:                       Lb, Ub, Deg : Integer;
9782:                       SVDTol : Float;
9783:                       B : TVector;
9784:                       V : TMMatrix); external 'dmath';
9785: { Unweighted polynomial regression by singular value decomposition }
9786: procedure WSVDPolFit(X, Y, S : TVector;
9787:                        Lb, Ub, Deg : Integer;

```

```

9788:           SVDTol      : Float;
9789:           B          : TVector;
9790:           V          : TMatrix); external 'dmath';
9791: { Weighted polynomial regression by singular value decomposition }
9792: procedure RegTest(Y, Ycalc : TVector;
9793:                      LbY, UbY : Integer;
9794:                      V          : TMatrix;
9795:                      LbV, UbV : Integer;
9796:                      var Test : TRegTest); external 'dmath';
9797: { Test of unweighted regression }
9798: procedure WRegTest(Y, Ycalc, S : TVector;
9799:                      LbY, UbY : Integer;
9800:                      V          : TMatrix;
9801:                      LbV, UbV : Integer;
9802:                      var Test : TRegTest); external 'dmath';
9803: { Test of weighted regression }
9804: { -----
9805: Nonlinear regression
9806: ----- }
9807: procedure SetOptAlgo(Algo : TOptAlgo); external 'dmath';
9808: { Sets the optimization algorithm for nonlinear regression }
9809: function GetOptAlgo : TOptAlgo; external 'dmath';
9810: { Returns the optimization algorithm }
9811: procedure SetMaxParam(N : Byte); external 'dmath';
9812: { Sets the maximum number of regression parameters for nonlinear regression }
9813: function GetMaxParam : Byte; external 'dmath';
9814: { Returns the maximum number of regression parameters for nonlinear regression }
9815: procedure SetParamBounds(I : Byte; ParamMin, ParamMax : Float); external 'dmath';
9816: { Sets the bounds on the I-th regression parameter }
9817: procedure GetParamBounds(I : Byte; var ParamMin, ParamMax:Float); external 'dmath';
9818: { Returns the bounds on the I-th regression parameter }
9819: procedure NLFit(RegFunc : TRegFunc;
9820:                      DerivProc : TDervProc;
9821:                      X, Y      : TVector;
9822:                      Lb, Ub    : Integer;
9823:                      MaxIter   : Integer;
9824:                      Tol       : Float;
9825:                      B          : TVector;
9826:                      FirstPar, LastPar : Integer;
9827:                      V          : TMatrix); external 'dmath';
9828: { Unweighted nonlinear regression }
9829: procedure WNLFit(RegFunc : TRegFunc;
9830:                      DerivProc : TDervProc;
9831:                      X, Y, S   : TVector;
9832:                      Lb, Ub    : Integer;
9833:                      MaxIter   : Integer;
9834:                      Tol       : Float;
9835:                      B          : TVector;
9836:                      FirstPar, LastPar : Integer;
9837:                      V          : TMatrix); external 'dmath';
9838: { Weighted nonlinear regression }
9839: procedure SetMCfile(FileName : String); external 'dmath';
9840: { Set file for saving MCMC simulations }
9841: procedure SimFit(RegFunc : TRegFunc;
9842:                      X, Y      : TVector;
9843:                      Lb, Ub    : Integer;
9844:                      B          : TVector;
9845:                      FirstPar, LastPar : Integer;
9846:                      V          : TMatrix); external 'dmath';
9847: { Simulation of unweighted nonlinear regression by MCMC }
9848: procedure WSimFit(RegFunc : TRegFunc;
9849:                      X, Y, S   : TVector;
9850:                      Lb, Ub    : Integer;
9851:                      B          : TVector;
9852:                      FirstPar, LastPar : Integer;
9853:                      V          : TMatrix); external 'dmath';
9854: { Simulation of weighted nonlinear regression by MCMC }
9855: procedure Nonlinear regression models
9856: ----- }
9857: procedure FracFit(X, Y      : TVector;
9858:                      Lb, Ub    : Integer;
9859:                      Deg1, Deg2 : Integer;
9860:                      ConstTerm : Boolean;
9861:                      MaxIter   : Integer;
9862:                      Tol       : Float;
9863:                      B          : TVector;
9864:                      V          : TMatrix); external 'dmath';
9865: { Unweighted fit of rational fraction }
9866: procedure WFractFit(X, Y, S : TVector;
9867:                      Lb, Ub    : Integer;
9868:                      Deg1, Deg2 : Integer;
9869:                      ConstTerm : Boolean;
9870:                      MaxIter   : Integer;
9871:                      Tol       : Float;

```

```

9877:           B      : TVector;
9878:           V      : TMatrix); external 'dmath';
9879: { Weighted fit of rational fraction }
9880:
9881: function FracFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9882: { Returns the value of the rational fraction at point X }
9883: procedure ExpFit(X, Y      : TVector;
9884:           Lb, Ub, Nexp : Integer;
9885:           ConsTerm   : Boolean;
9886:           MaxIter    : Integer;
9887:           Tol       : Float;
9888:           B         : TVector;
9889:           V         : TMatrix); external 'dmath';
9890: { Unweighted fit of sum of exponentials }
9891: procedure WExpFit(X, Y, S      : TVector;
9892:           Lb, Ub, Nexp : Integer;
9893:           ConsTerm   : Boolean;
9894:           MaxIter    : Integer;
9895:           Tol       : Float;
9896:           B         : TVector;
9897:           V         : TMatrix); external 'dmath';
9898: { Weighted fit of sum of exponentials }
9899: function ExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9900: { Returns the value of the regression function at point X }
9901: procedure IncExpFit(X, Y      : TVector;
9902:           Lb, Ub      : Integer;
9903:           ConsTerm   : Boolean;
9904:           MaxIter    : Integer;
9905:           Tol       : Float;
9906:           B         : TVector;
9907:           V         : TMatrix); external 'dmath';
9908: { Unweighted fit of model of increasing exponential }
9909: procedure WIIncExpFit(X, Y, S      : TVector;
9910:           Lb, Ub      : Integer;
9911:           ConsTerm   : Boolean;
9912:           MaxIter    : Integer;
9913:           Tol       : Float;
9914:           B         : TVector;
9915:           V         : TMatrix); external 'dmath';
9916: { Weighted fit of increasing exponential }
9917: function IncExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9918: { Returns the value of the regression function at point X }
9919: procedure ExpLinFit(X, Y      : TVector;
9920:           Lb, Ub      : Integer;
9921:           MaxIter    : Integer;
9922:           Tol       : Float;
9923:           B         : TVector;
9924:           V         : TMatrix); external 'dmath';
9925: { Unweighted fit of the "exponential + linear" model }
9926: procedure WExpLinFit(X, Y, S      : TVector;
9927:           Lb, Ub      : Integer;
9928:           MaxIter    : Integer;
9929:           Tol       : Float;
9930:           B         : TVector;
9931:           V         : TMatrix); external 'dmath';
9932: { Weighted fit of the "exponential + linear" model }
9933:
9934: function ExpLinFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9935: { Returns the value of the regression function at point X }
9936: procedure MichFit(X, Y      : TVector;
9937:           Lb, Ub      : Integer;
9938:           MaxIter    : Integer;
9939:           Tol       : Float;
9940:           B         : TVector;
9941:           V         : TMatrix); external 'dmath';
9942: { Unweighted fit of Michaelis equation }
9943: procedure WMichFit(X, Y, S      : TVector;
9944:           Lb, Ub      : Integer;
9945:           MaxIter    : Integer;
9946:           Tol       : Float;
9947:           B         : TVector;
9948:           V         : TMatrix); external 'dmath';
9949: { Weighted fit of Michaelis equation }
9950: function MichFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9951: { Returns the value of the Michaelis equation at point X }
9952: procedure MintFit(X, Y      : TVector;
9953:           Lb, Ub      : Integer;
9954:           MintVar   : TMintVar;
9955:           Fit_S0    : Boolean;
9956:           MaxIter    : Integer;
9957:           Tol       : Float;
9958:           B         : TVector;
9959:           V         : TMatrix); external 'dmath';
9960: { Unweighted fit of the integrated Michaelis equation }
9961: procedure WMintFit(X, Y, S      : TVector;
9962:           Lb, Ub      : Integer;
9963:           MintVar   : TMintVar;
9964:           Fit_S0    : Boolean;
9965:           MaxIter    : Integer;

```

```

9966:           Tol      : Float;
9967:           B       : TVector;
9968:           V       : TMatrix); external 'dmath';
9969: { Weighted fit of the integrated Michaelis equation }
9970: function MintFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9971: { Returns the value of the integrated Michaelis equation at point X }
9972: procedure HillFit(X, Y : TVector;
9973:           Lb, Ub : Integer;
9974:           MaxIter : Integer;
9975:           Tol      : Float;
9976:           B       : TVector;
9977:           V       : TMatrix); external 'dmath';
9978: { Unweighted fit of Hill equation }
9979: procedure WHillFit(X, Y, S : TVector;
9980:           Lb, Ub : Integer;
9981:           MaxIter : Integer;
9982:           Tol      : Float;
9983:           B       : TVector;
9984:           V       : TMatrix); external 'dmath';
9985: { Weighted fit of Hill equation }
9986: function HillFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9987: { Returns the value of the Hill equation at point X }
9988: procedure LogiFit(X, Y : TVector;
9989:           Lb, Ub : Integer;
9990:           ConsTerm : Boolean;
9991:           General : Boolean;
9992:           MaxIter : Integer;
9993:           Tol      : Float;
9994:           B       : TVector;
9995:           V       : TMatrix); external 'dmath';
9996: { Unweighted fit of logistic function }
9997: procedure WLogiFit(X, Y, S : TVector;
9998:           Lb, Ub : Integer;
9999:           ConsTerm : Boolean;
10000:          General : Boolean;
10001:          MaxIter : Integer;
10002:          Tol      : Float;
10003:          B       : TVector;
10004:          V       : TMatrix); external 'dmath';
10005: { Weighted fit of logistic function }
10006: function LogiFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10007: { Returns the value of the logistic function at point X }
10008: procedure PKFit(X, Y : TVector;
10009:           Lb, Ub : Integer;
10010:          MaxIter : Integer;
10011:          Tol      : Float;
10012:          B       : TVector;
10013:          V       : TMatrix); external 'dmath';
10014: { Unweighted fit of the acid-base titration curve }
10015: procedure WPKFit(X, Y, S : TVector;
10016:           Lb, Ub : Integer;
10017:           MaxIter : Integer;
10018:           Tol      : Float;
10019:           B       : TVector;
10020:           V       : TMatrix); external 'dmath';
10021: { Weighted fit of the acid-base titration curve }
10022: function PKFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10023: { Returns the value of the acid-base titration function at point X }
10024: procedure PowFit(X, Y : TVector;
10025:           Lb, Ub : Integer;
10026:           MaxIter : Integer;
10027:           Tol      : Float;
10028:           B       : TVector;
10029:           V       : TMatrix); external 'dmath';
10030: { Unweighted fit of power function }
10031: procedure WPowFit(X, Y, S : TVector;
10032:           Lb, Ub : Integer;
10033:           MaxIter : Integer;
10034:           Tol      : Float;
10035:           B       : TVector;
10036:           V       : TMatrix); external 'dmath';
10037: { Weighted fit of power function }
10038: function PowFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10040: { Returns the value of the power function at point X }
10041: procedure GammaFit(X, Y : TVector;
10042:           Lb, Ub : Integer;
10043:           MaxIter : Integer;
10044:           Tol      : Float;
10045:           B       : TVector;
10046:           V       : TMatrix); external 'dmath';
10047: { Unweighted fit of gamma distribution function }
10048: procedure WGammaFit(X, Y, S : TVector;
10049:           Lb, Ub : Integer;
10050:           MaxIter : Integer;
10051:           Tol      : Float;
10052:           B       : TVector;
10053:           V       : TMatrix); external 'dmath';
10054: { Weighted fit of gamma distribution function }

```

```

10055: function GammaFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10056: { Returns the value of the gamma distribution function at point X }
10057: { -----
10058:   Principal component analysis
10059:   -----
10060: procedure VecMean(X           : TMatrix;
10061:                      Lb, Ub, Nvar : Integer;
10062:                      M           : TVector); external 'dmath';
10063: { Computes the mean vector M from matrix X }
10064: procedure VecSD(X           : TMatrix;
10065:                      Lb, Ub, Nvar : Integer;
10066:                      M, S        : TVector); external 'dmath';
10067: { Computes the vector of standard deviations S from matrix X }
10068: procedure MatVarCov(X          : TMatrix;
10069:                        Lb, Ub, Nvar : Integer;
10070:                        M           : TVector;
10071:                        V           : TMatrix); external 'dmath';
10072: { Computes the variance-covariance matrix V from matrix X }
10073: procedure MatCorrel(V         : TMatrix;
10074:                        Nvar : Integer;
10075:                        R           : TMatrix); external 'dmath';
10076: { Computes the correlation matrix R from the var-cov matrix V }
10077: procedure PCA(R            : TMatrix;
10078:                      Nvar : Integer;
10079:                      Lambda : TVector;
10080:                      C, Rc : TMatrix); external 'dmath';
10081: { Performs a principal component analysis of the correlation matrix R }
10082: procedure ScaleVar(X          : TMatrix;
10083:                        Lb, Ub, Nvar : Integer;
10084:                        M, S        : TVector;
10085:                        Z           : TMatrix); external 'dmath';
10086: { Scales a set of variables by subtracting means and dividing by SD's }
10087: procedure PrinFac(Z         : TMatrix;
10088:                        Lb, Ub, Nvar : Integer;
10089:                        C, F        : TMatrix); external 'dmath';
10090: { Computes principal factors }
10091: { -----
10092:   Strings
10093:   -----
10094: function LTrim(S : String) : String; external 'dmath';
10095: { Removes leading blanks }
10096: function RTrim(S : String) : String; external 'dmath';
10097: { Removes trailing blanks }
10098: function Trim(S : String) : String; external 'dmath';
10099: { Removes leading and trailing blanks }
10100: function StrChar(N : Byte; C : Char) : String; external 'dmath';
10101: { Returns a string made of character C repeated N times }
10102: function RFill(S : String; L : Byte) : String; external 'dmath';
10103: { Completes string S with trailing blanks for a total length L }
10104: function LFill(S : String; L : Byte) : String; external 'dmath';
10105: { Completes string S with leading blanks for a total length L }
10106: function CFill(S : String; L : Byte) : String; external 'dmath';
10107: { Centers string S on a total length L }
10108: function Replace(S : String; C1, C2 : Char) : String; external 'dmath';
10109: { Replaces in string S all the occurrences of C1 by C2 }
10110: function Extract(S : String; var Index : Byte; Delim : Char) : String; external 'dmath';
10111: { Extracts a field from a string }
10112: procedure Parse(S : String; Delim:Char; Field:TStrVector; var N:Byte); external 'dmath';
10113: { Parses a string into its constitutive fields }
10114: procedure SetFormat(NumLength,MaxDec:Integer;FloatPoint,NSZero:Bool); external 'dmath';
10115: { Sets the numeric format }
10116: function FloatToStr(X : Float) : String; external 'dmath';
10117: { Converts a real to a string according to the numeric format }
10118: function IntToStr(N : LongInt) : String; external 'dmath';
10119: { Converts an integer to a string }
10120: function CompStr(Z : Complex) : String; external 'dmath';
10121: { Converts a complex number to a string }
10122: {$_IFDEF DELPHI}
10123: function StrDec(S : String) : String; external 'dmath';
10124: { Set decimal separator to the symbol defined in SysUtils }
10125: function IsNumeric(var S : String; var X : Float) : Boolean; external 'dmath';
10126: { Test if a string represents a number and returns it in X }
10127: function ReadNumFromEdit(Edit : TEdit) : Float; external 'dmath';
10128: { Reads a floating point number from an Edit control }
10129: procedure WriteNumToFile(var F : Text; X : Float); external 'dmath';
10130: { Writes a floating point number in a text file }
10131: {$_ENDIF}
10132: { -----
10133:   BGI / Delphi graphics
10134:   -----
10135: function InitGraphics
10136: {$_IFDEF DELPHI}
10137: (Width, Height : Integer) : Boolean;
10138: {$_ELSE}
10139: (Pilot, Mode : Integer; BGIPath : String) : Boolean; {$_ENDIF} external 'dmath';
10140: { Enters graphic mode }
10141: procedure SetWindow({$_IFDEF DELPHI}Canvas : TCanvas;{$_ENDIF}
10142:                         X1, X2, Y1,Y2 : Integer; GraphBorder:Boolean); external 'dmath';
10143: { Sets the graphic window }

```

```

10144: procedure SetOxScale(Scale           : TScale;
10145:                           OxMin, OxMax, OxStep : Float); external 'dmath';
10146: { Sets the scale on the Ox axis }
10147: procedure SetOyScale(Scale           : TScale;
10148:                           OyMin, OyMax, OyStep : Float); external 'dmath';
10149: { Sets the scale on the Oy axis }
10150: procedure GetOxScale(var Scale       : TScale;
10151:                           var OxMin, OxMax, OxStep : Float); external 'dmath';
10152: { Returns the scale on the Ox axis }
10153: procedure GetOyScale(var Scale       : TScale;
10154:                           var OyMin, OyMax, OyStep : Float); external 'dmath';
10155: { Returns the scale on the Oy axis }
10156: procedure SetGraphTitle>Title : String); external 'dmath'; { Sets the title for the graph }
10157: procedure SetOxTitle>Title : String); external 'dmath'; { Sets the title for the Ox axis }
10158: procedure SetOyTitle>Title : String); external 'dmath'; { Sets the title for the Oy axis }
10159: function GetGraphTitle : String; external 'dmath'; { Returns the title for the graph }
10160: function GetOxTitle : String; external 'dmath'; { Returns the title for the Ox axis }
10161: function GetOyTitle : String; external 'dmath'; { Returns the title for the Oy axis }
10162: {$IFDEF DELPHI}
10163: procedure SetTitleFont(FontIndex, Width, Height : Integer); external 'dmath';
10164: { Sets the font for the main graph title }
10165: procedure SetOxFont(FontIndex, Width, Height : Integer); external 'dmath';
10166: { Sets the font for the Ox axis (title and labels) }
10167: procedure SetOyFont(FontIndex, Width, Height : Integer); external 'dmath';
10168: { Sets the font for the Oy axis (title and labels) }
10169: procedure SetLgdFont(FontIndex, Width, Height : Integer); external 'dmath';
10170: { Sets the font for the legends }
10171: procedure SetClipping(Clip : Boolean); external 'dmath';
10172: { Determines whether drawings are clipped at the current viewport
10173:   boundaries, according to the value of the Boolean parameter Clip }
10174: {$ENDIF}
10175: procedure PlotOxAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10176: { Plots the horizontal axis }
10177: procedure PlotOyAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10178: { Plots the vertical axis }
10179: procedure PlotGrid{$IFDEF DELPHI}Canvas:TCanvas; Grid:TGrid){$ENDIF}; external 'dmath';
10180: { Plots a grid on the graph }
10181: procedure WriteGraphTitle{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10182: { Writes the title of the graph }
10183: procedure SetMaxCurv(NCurv : Byte); external 'dmath';
10184: { Sets the maximum number of curves and re-initializes their parameters }
10185: procedure SetPointParam
10186: {$IFDEF DELPHI}
10187: (CurvIndex, Symbol, Size : Integer; Color : TColor);
10188: {$ELSE}
10189: (CurvIndex, Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10190: { Sets the point parameters for curve # CurvIndex }
10191: procedure SetLineParam
10192: {$IFDEF DELPHI}
10193: (CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor);
10194: {$ELSE}
10195: (CurvIndex, Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10196: { Sets the line parameters for curve # CurvIndex }
10197: procedure SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10198: { Sets the legend for curve # CurvIndex }
10199: procedure SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10200: { Sets the step for curve # CurvIndex }
10201: function GetMaxCurv : Byte; external 'dmath'; { Returns the maximum number of curves }
10202: procedure GetPointParam
10203: {$IFDEF DELPHI}
10204: (CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor);
10205: {$ELSE}
10206: (CurvIndex : Integer; var Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10207: { Returns the point parameters for curve # CurvIndex }
10208: procedure GetLineParam
10209: {$IFDEF DELPHI}
10210: (CurvIndex : Integer; var Style : TPenStyle; var Width : Integer; var Color : TColor);
10211: {$ELSE}
10212: (CurvIndex : Integer; var Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10213: { Returns the line parameters for curve # CurvIndex }
10214: function GetCurvLegend(CurvIndex : Integer) : String; external 'dmath';
10215: { Returns the legend for curve # CurvIndex }
10216: function GetCurvStep(CurvIndex : Integer) : Integer; external 'dmath';
10217: { Returns the step for curve # CurvIndex }
10218: {$IFDEF DELPHI}
10219: procedure PlotPoint(Canvas      : TCanvas;
10220:                         X, Y       : Float; CurvIndex : Integer); external 'dmath';
10221: {$ELSE}
10222: procedure PlotPoint(Xp, Yp, CurvIndex : Integer); external 'dmath';
10223: {$ENDIF}
10224: { Plots a point on the screen }
10225: procedure PlotCurve{$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10226: (X, Y           : TVector;
10227:   Lb, Ub, CurvIndex : Integer); external 'dmath';
10228: { Plots a curve }
10229: procedure PlotCurveWithErrorBars{$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10230: (X, Y, S          : TVector;
10231:   Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10232: { Plots a curve with error bars }

```

```

10233: procedure PlotFunc({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10234:                      Func          : TFunc;
10235:                      Xmin, Xmax   : Float;
10236:                      {$IFDEF DELPHI}Npt    : Integer;{$ENDIF}
10237:                      CurvIndex    : Integer); external 'dmath';
10238: { Plots a function }
10239: procedure WriteLegend({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10240:                         NCurv        : Integer;
10241:                         ShowPoints, ShowLines : Boolean); external 'dmath';
10242: { Writes the legends for the plotted curves }
10243: procedure ConRec({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10244:                      Nx, Ny, Nc   : Integer;
10245:                      X, Y, Z      : TVector;
10246:                      F             : TMatrix); external 'dmath';
10247: { Contour plot }
10248: function Xpixel(X : Float):Integer; external 'dmath'; { Converts user abscissa X to screen coordinate }
10249: function Ypixel(Y : Float):Integer; external 'dmath'; { Converts user ordinate Y to screen coordinate }
10250: function Xuser(X : Integer):Float; external 'dmath'; { Converts screen coordinate X to user abscissa }
10251: function Yuser(Y : Integer):Float; external 'dmath'; { Converts screen coordinate Y to user ordinate }
10252: {$IFNDEF DELPHI}
10253: procedure LeaveGraphics; external 'dmath';
10254: { Quits graphic mode }
10255: {$ENDIF}
10256: { -----
10257:  LaTeX graphics
10258:  -----
10259: function TeX_InitGraphics(FileName      : String; PgWidth, PgHeight : Integer;
10260:                           Header       : Boolean); external 'dmath';
10261: { Initializes the LaTeX file }
10262: procedure TeX_SetWindow(X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean); external 'dmath';
10263: { Sets the graphic window }
10264: procedure TeX_LeaveGraphics(Footer : Boolean); external 'dmath'; { Close the LaTeX file }
10265: procedure TeX_SetOxScale(Scale : TScale; OxMin, OxMax, OxStep : Float); external 'dmath';
10266: { Sets the scale on the Ox axis }
10267: procedure TeX_SetOyScale(Scale : TScale; OyMin, OyMax, OyStep : Float); external 'dmath';
10268: { Sets the scale on the Oy axis }
10269: procedure TeX_SetGraphTitle>Title : String); external 'dmath'; { Sets the title for the graph }
10270: procedure TeX_SetOxTitle>Title : String); external 'dmath'; { Sets the title for the Ox axis }
10271: procedure TeX_SetOyTitle>Title : String); external 'dmath'; { Sets the title for the Oy axis }
10272: procedure TeX_PlotOxAxis; external 'dmath'; { Plots the horizontal axis }
10273: procedure TeX_PlotOyAxis; external 'dmath'; { Plots the vertical axis }
10274: procedure TeX_PlotGrid(Grid : TGrid); external 'dmath'; { Plots a grid on the graph }
10275: procedure TeX_WriteGraphTitle; external 'dmath'; Writes the title of the graph
10276: procedure TeX_SetMaxCurv(NCurv : Byte); external 'dmath';
10277: { Sets the maximum number of curves and re-initializes their parameters }
10278: procedure TeX_SetPointParam(CurvIndex, Symbol, Size : Integer); external 'dmath';
10279: { Sets the point parameters for curve # CurvIndex }
10280: procedure TeX_SetLineParam(CurvIndex, Style : Integer;
10281:                             Width : Float; Smooth : Boolean); external 'dmath';
10282: { Sets the line parameters for curve # CurvIndex }
10283: procedure TeX_SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10284: { Sets the legend for curve # CurvIndex }
10285: procedure TeX_SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10286: { Sets the step for curve # CurvIndex }
10287: procedure TeX_PlotCurve(X, Y : TVector; Lb, Ub, CurvIndex : Integer); external 'dmath';
10288: { Plots a curve }
10289: procedure TeX_PlotCurveWithErrorBars(X, Y, S : TVector;
10290:                                       Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10291: { Plots a curve with error bars }
10292: procedure TeX_PlotFunc(Func : TFunc; X1, X2 : Float;
10293:                          Npt : Integer; CurvIndex : Integer); external 'dmath';
10294: { Plots a function }
10295: procedure TeX_WriteLegend(NCurv : Integer; ShowPoints, ShowLines : Boolean); external 'dmath';
10296: { Writes the legends for the plotted curves }
10297: procedure TeX_ConRec(Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix); external 'dmath';
10298: { Contour plot }
10299: function Xcm(X : Float) : Float; external 'dmath'; { Converts user coordinate X to cm }
10300: function Ycm(Y : Float) : Float; external 'dmath'; { Converts user coordinate Y to cm }
10301:
10302: //*****unit uPSI_SynPdf;
10303: Function RawUTF8ToPDFString( const Value : RawUTF8 ) : PDFString
10304: Function _DateToString( ADate : TDateTime ) : TPdfDate
10305: Function _PDFDateToDate( const AText : TPdfDate ) : TDateTime
10306: Function PdfRect( Left, Top, Right, Bottom : Single ) : TPdfRect;
10307: Function PdfRectL( const Box : TPdfBox ) : TPdfRect;
10308: Function PdfBox( Left, Top, Width, Height : Single ) : TPdfBox
10309: //Function _GetCharCount( Text : PAnsiChar ) : integer
10310: //Procedure L2R( W : PWideChar; L : integer)
10311: Function PdfCoord( MM : single ) : integer
10312: Function CurrentPrinterPageSize : TPDFPageSize
10313: Function CurrentPrinterRes : TPoint
10314: Procedure GDICommentBookmark( MetaHandle : HDC; const aBookmarkName : RawUTF8 )
10315: Procedure GDICommentOutline( MetaHandle : HDC; const aTitle : RawUTF8; aLevel : Integer )
10316: Procedure GDICommentLink( MetaHandle : HDC; const aBookmarkName : RawUTF8; const aRect : TRect )
10317: Const ('Uspl0','String 'uspl0.dll
10318: AddTypes('TScriptState_enum', '( r0, r1, r2, r3, r4, fOverrideDirection, fInhibitSymSwap,
10319: 'fcharShape, fDigitSubstitute,fInhibitLigate,fDisplayZWG, fArabicNumContext, fGcpClusters )
10320: AddTypes('TScriptState_set', 'set of TScriptState_enum
10321: //*****

```

```

10322:
10323: procedure SIRегистер_PMrand(CL: TPSPascalCompiler); //ParkMiller
10324: begin
10325:   Procedure PMrandomize( I : word)
10326:   Function PMrandom : longint
10327:   Function Rrand : extended
10328:   Function Irand( N : word) : word
10329:   Function Brand( P : extended) : boolean
10330:   Function Nrand : extended
10331: end;
10332:
10333: procedure SIRегистер_Spring_Cryptography_Utils(CL: TPSPascalCompiler);
10334: begin
10335:   Function Endian( x : LongWord) : LongWord
10336:   Function Endian64( x : Int64) : Int64
10337:   Function spRol( x : LongWord; y : Byte) : LongWord
10338:   Function spRor( x : LongWord; y : Byte) : LongWord
10339:   Function Ror64( x : Int64; y : Byte) : Int64
10340: end;
10341:
10342: procedure SIRегистер_MapReader(CL: TPSPascalCompiler);
10343: begin
10344:   Procedure ClearModules
10345:   Procedure ReadMapFile( Fname : string)
10346:   Function AddressInfo( Address : dword) : string
10347: end;
10348:
10349: procedure SIRегистер_LibTar(CL: TPSPascalCompiler);
10350: begin
10351:   TTarPermission', '( tpReadByOwner, tpWriteByOwner, tpExecuteByOwn'
10352:   +'ner, tpReadByGroup, tpWriteByGroup, tpExecuteByGroup, tpReadByOther, tpWri'
10353:   +'teByOther, tpExecuteByOther )
10354:   TTarPermissions', 'set of TTarPermission
10355:   TFileType', '( ftNormal, ftLink, ftSymbolicLink, ftCharacter, ft'
10356:   +'Block, ftDirectory, ftFifo, ftContiguous, ftDumpDir, ftMultiVolume, ftVolumeHeader,
10357:   TTarMode', '( tmSetUid, tmSetGid, tmSaveText )
10358:   TTarModes', 'set of TTarMode
10359:   TTarDirRec', 'record Name : STRING; Size : INT64; DateTime : TDa'
10360:   +'teTime; Permissions : TTarPermissions; FileType : TFileType; LinkName : ST'
10361:   +'RING; UID : INTEGER; GID : INTEGER; UserName : STRING; GroupName : STRING;'
10362:   +' ChecksumOK : BOOLEAN; Mode : TTarModes; Magic : STRING; MajorDevNo : INTE'
10363:   +'GER; MinorDevNo : INTEGER; FilePos : INT64; end
10364:   SIRегистер_TTarArchive(CL);
10365:   SIRегистер_TTarWriter(CL);
10366:   Function PermissionString( Permissions : TTarPermissions ) : STRING
10367:   Function ConvertFilename( Filename : STRING ) : STRING
10368:   Function FileTimeGMT( FileName : STRING ) : TDateTime;
10369:   Function FileTimeGMT1( SearchRec : TSearchRec ) : TDateTime;
10370:   Procedure ClearDirRec( var DirRec : TTarDirRec )
10371: end;
10372:
10373:
10374: //*****unit uPSI_TlHelp32;
10375: procedure SIRегистер_TlHelp32(CL: TPSPascalCompiler);
10376: begin
10377:   Const('MAX_MODULE_NAME32','LongInt'( 255 );
10378:   Function CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD ) : THandle
10379:   Const('TH32CS_SNAPHEAPLIST','LongWord( $00000001 );
10380:   Const('TH32CS_SNAPPROCESS','LongWord').SetUInt( $00000002 );
10381:   Const('TH32CS_SNAPTHREAD','LongWord').SetUInt( $00000004 );
10382:   Const('TH32CS_SNAPMODULE','LongWord').SetUInt( $00000008 );
10383:   Const('TH32CS_INHERIT','LongWord').SetUInt( $80000000 );
10384:   tagHEAPLIST32', 'record dwSize:DWORD;th32ProcessID:DWORD;th32HeapID:DWORD;dwFlags:DWORD;end';
10385:   AddTypeS('HEAPLIST32', 'tagHEAPLIST32
10386:   AddTypeS('THeapList32', 'tagHEAPLIST32
10387:   Const('HF32_DEFAULT','LongInt'( 1 );
10388:   Const('HF32_SHARED','LongInt'( 2 );
10389:   Function Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32 ) : BOOL
10390:   Function Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32 ) : BOOL
10391:   AddTypeS('tagHEAPENTRY32', 'record dwSize : DWORD; hHandle : THandle; dwAd'
10392:   +'dress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
10393:   +'dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end
10394:   AddTypeS('HEAPENTRY32', 'tagHEAPENTRY32
10395:   AddTypeS('TheapEntry32', 'tagHEAPENTRY32
10396:   Const('LF32_FIXED','LongWord').SetUInt( $00000001 );
10397:   Const('LF32_FREE','LongWord').SetUInt( $00000002 );
10398:   Const('LF32_MOVEABLE','LongWord').SetUInt( $00000004 );
10399:   Function Heap32First( var lphe : TheapEntry32; th32ProcessID, th32HeapID : DWORD ) : BOOL
10400:   Function Heap32Next( var lphe : TheapEntry32 ) : BOOL
10401:   DWORD; var lpNumberOfBytesRead : DWORD) : BOOL
10402:   AddTypeS('tagTHREADENTRY32', 'record dwSize : DWORD; cntUsage : DWORD; th3'
10403:   +'2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt'
10404:   +'APri : Longint; dwFlags : DWORD; end
10405:   AddTypeS('THREADENTRY32', 'tagTHREADENTRY32
10406:   AddTypeS('TThreadEntry32', 'tagTHREADENTRY32
10407:   Function Thread32First( hSnapshot : THandle; var lppe : TThreadEntry32 ) : BOOL
10408:   Function Thread32Next( hSnapshot : THandle; var lppe : TThreadEntry32 ) : BOOL
10409: end;
10410: Const('EW_RESTARTWINDOWS','LongWord').SetUInt( $0042 );

```

```

10411: Const ('EW_REBOOTSYSTEM', 'LongWord( $0043);
10412: Const ('EW_EXITANDEXECAPP', 'LongWord( $0044);
10413: Const ('ENDSESSION_LOGOFF','LongWord').SetUInt( DWORD ( $80000000 ) );
10414: Const ('EWX_LOGOFF','LongInt'( 0);
10415: Const ('EWX_SHUTDOWN','LongInt'( 1);
10416: Const ('EWX_REBOOT','LongInt'( 2);
10417: Const ('EWX_FORCE','LongInt'( 4);
10418: Const ('EWX_POWEROFF','LongInt'( 8);
10419: Const ('EWX_FORCEIFHUNG','LongWord').SetUInt( $10);
10420: Function GET_APPCOMMAND_LPARAM( const lParam : LongInt ) : Shortint;
10421: Function GET_DEVICE_LPARAM( const lParam : LongInt ) : Word;
10422: Function GET_MOUSEKEY_LPARAM( const lParam : LongInt ) : Word;
10423: Function GET_FLAGS_LPARAM( const lParam : LongInt ) : Word;
10424: Function GET_KEYSTATE_LPARAM( const lParam : Longint ) : Word;
10425: Function GetWindowWord( hWnd : HWND; nIndex : Integer ) : Word;
10426: Function SetWindowWord( hWnd : HWND; nIndex : Integer; wNewWord : Word ) : Word;
10427: Function GetWindowLong( hWnd : HWND; nIndex : Integer ) : Longint;
10428: Function SetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : Longint;
10429: Function GetClassWord( hWnd : HWND; nIndex : Integer ) : Word;
10430: Function SetClassWord( hWnd : HWND; nIndex : Integer; wNewWord : Word ) : Word;
10431: Function GetClassLong( hWnd : HWND; nIndex : Integer ) : DWORD;
10432: Function SetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : DWORD;
10433: Function GetDesktopWindow : HWND;
10434: Function GetParent( hWnd : HWND ) : HWND;
10435: Function SetParent( hWndChild, hWndNewParent : HWND ) : HWND;
10436: Function GetTopWindow( hWnd : HWND ) : HWND;
10437: Function GetNextWindow( hWnd : HWND; uCmd : UINT ) : HWND;
10438: Function GetWindow( hWnd : HWND; uCmd : UINT ) : HWND;
10439: //Delphi DFM
10440: Function LoadDFMFile2Strings( const AFile:string; AStrings:TStrings; var WasText:boolean):integer;
10441: Function SaveStrings2DFMFile( AStrings : TStrings; const Afile : string ) : integer;
10442: procedure GetHighlighters( AOwner: TComponent; AHighlighters: TStringList; AppendToList: boolean );
10443: function GetHighlightersFilter(AHighlighters: TStringList): string;
10444: function GetHighlighterFromFileExt(AHighlighters: TStringList; Extension: string): TSynCustomHighlighter;
10445: Function ShowOwnedPopups( hWnd : HWND; fShow : BOOL ) : BOOL;
10446: Function OpenIcon( hWnd : HWND ) : BOOL;
10447: Function CloseWindow( hWnd : HWND ) : BOOL;
10448: Function MoveWindow( hWnd : HWND; X, Y, nWidth, nHeight : Integer; bRepaint : BOOL ) : BOOL;
10449: Function SetWindowPos( hWnd : HWND; hWndInsertAfter:HWND; X,Y,cx,cy : Integer; uFlags : UINT ) : BOOL;
10450: Function IsWindowVisible( hWnd : HWND ) : BOOL;
10451: Function IsIconic( hWnd : HWND ) : BOOL;
10452: Function AnyPopup : BOOL;
10453: Function BringWindowToFront( hWnd : HWND ) : BOOL;
10454: Function IsZoomed( hWnd : HWND ) : BOOL;
10455: Function IsWindow( hWnd : HWND ) : BOOL;
10456: Function IsMenu( hMenu : HMENU ) : BOOL;
10457: Function IsChild( hWndParent, hWnd : HWND ) : BOOL;
10458: Function DestroyWindow( hWnd : HWND ) : BOOL;
10459: Function ShowWindow( hWnd : HWND; nCmdShow : Integer ) : BOOL;
10460: Function AnimateWindow( hWnd : HWND; dwTime : DWORD; dwFlags : DWORD ) : BOOL;
10461: Function ShowWindowAsync( hWnd : HWND; nCmdShow : Integer ) : BOOL;
10462: Function FlashWindow( hWnd : HWND; bInvert : BOOL ) : BOOL;
10463: Function IsWindowUnicode( hWnd : HWND ) : BOOL;
10464: Function EnableWindow( hWnd : HWND; bEnable : BOOL ) : BOOL;
10465: Function IsWindowEnabled( hWnd : HWND ) : BOOL;
10466:
10467: procedure SIRegister_IDECmdLine(CL: TPPascalCompiler);
10468: begin
10469:   const ('ShowSetupDialogOptLong','String '--setup
10470:   PrimaryConfPathOptLong','String '--primary-config-path=
10471:   PrimaryConfPathOptShort','String '--pcp=
10472:   SecondaryConfPathOptLong','String '--secondary-config-path=
10473:   SecondaryConfPathOptShort','String '--scp=
10474:   NoSplashScreenOptLong','String '--no-splash-screen
10475:   NoSplashScreenOptShort','String '--nsc
10476:   StartedByStartLazarusOpt','String '--started-by-startlazarus
10477:   SkipLastProjectOpt','String '--skip-last-project
10478:   DebugLogOpt','String '--debug-log=
10479:   DebugLogOptEnable','String '--debug-enable=
10480:   LanguageOpt','String '--language=
10481:   LazarusDirOpt','String '--lazarusdir=
10482:   Procedure ParseCommandLine(aCmdLineParams:TStrings;out IDEPid:Int;out ShowSplashScreen:boolean);
10483:   Function GetCommandLineParameters( aCmdLineParams : TStrings; isStartLazarus:Boolean ) : string;
10484:   Function ExtractPrimaryConfigPath( aCmdLineParams : TStrings ) : string;
10485:   Function IsHelpRequested : Boolean;
10486:   Function IsVersionRequested : boolean;
10487:   Function GetLanguageSpecified : string;
10488:   Function ParamIsOption( ParamIndex : integer; const Option : string ) : boolean;
10489:   Function ParamIsOptionPlusValue(ParamIndex:integer;const Option:string;out AValue:string):bool;
10490:   Procedure ParseNoGuiCmdLineParams;
10491:   Function ExtractCmdLineFilenames : TStrings;
10492: end;
10493:
10494:
10495: procedure SIRegister_LazFileUtils(CL: TPPascalCompiler);
10496: begin
10497:   Function CompareFilenames( const Filenamel, Filename2 : string ) : integer;
10498:   Function CompareFilenamesIgnoreCase( const Filenamel, Filename2 : string ) : integer;
10499:   Function CompareFileExt( const Filename, Ext : string; CaseSensitive : boolean ) : integer;

```

```

10500: Function CompareFileExt1( const Filename, Ext : string ) : integer;
10501: Function CompareFilenameStarts( const Filenamel, Filename2 : string ) : integer
10502: Function CompareFilenames(Filenamel:PChar;Len1:integer; Filename2:PChar;Len2:integer):integer
10503: Function CompareFilenamesP( Filenamel, Filename2 : PChar; IgnoreCase : boolean ) : integer
10504: Function DirPathExists( DirectoryName : string ) : boolean
10505: Function DirectoryIsWritable( const DirectoryName : string ) : boolean
10506: Function ExtractFileNameOnly( const AFilename : string ) : string
10507: Function FilenameIsAbsolute( const Thefilename : string ) : boolean
10508: Function FilenameIsWinAbsolute( const Thefilename : string ) : boolean
10509: Function FilenameIsUnixAbsolute( const Thefilename : string ) : boolean
10510: Function ForceDirectory( DirectoryName : string ) : boolean
10511: Procedure CheckIffileIsExecutable( const AFilename : string )
10512: Procedure CheckIfFileIsSymlink( const AFilename : string )
10513: Function FileIsText( const AFilename : string ) : boolean
10514: Function FileIsText2( const AFilename : string; out FileReadable : boolean ) : boolean
10515: Function FilenameIsTrimmed( const Thefilename : string ) : boolean
10516: Function FilenameIsTrimmed2( StartPos : PChar; NameLen : integer ) : boolean
10517: Function TrimFilename( const AFilename : string ) : string
10518: Function ResolveDots( const AFilename : string ) : string
10519: Procedure ForcePathDelims( var FileName : string )
10520: Function GetForcedPathDelims( const FileName : string ) : String
10521: Function CleanAndExpandFilename( const Filename : string ) : string
10522: Function CleanAndExpandDirectory( const Filename : string ) : string
10523: Function TrimAndExpandFilename( const Filename : string; const BaseDir : string ) : string
10524: Function TrimAndExpandDirectory( const Filename : string; const BaseDir : string ) : string
10525: Function TryCreateRelativePath(const Dest,Source:String; UsePointDirectory:bool;
  AlwaysRequireSharedBaseFolder : Boolean; out RelPath : String) : Boolean
10526: Function CreateRelativePath( const Filename,BaseDirectory:string; UsePointDirectory:boolean;
  AlwaysRequireSharedBaseFolder: Boolean) : string
10527: Function FileIsInPath( const Filename, Path : string ) : boolean
10528: Function AppendPathDelim( const Path : string ) : string
10529: Function ChompPathDelim( const Path : string ) : string
10530: Function CreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string ) : string
10531: Function CreateRelativeSearchPath( const SearchPath, BaseDirectory : string ) : string
10532: Function MinimizeSearchPath( const SearchPath : string ) : string
10533: Function FindPathInSearchPath(APath:PChar;APathLen:int;SearchPath:PChar;SearchPathLen:int):PChar;
(*Function FileExistsUTF8( const FileName : string ) : boolean
10535: Function FileAgeUTF8( const FileName : string ) : Longint
10536: Function DirectoryExistsUTF8( const Directory : string ) : Boolean
10537: Function ExpandFileNameUTF8( const FileName : string; BaseDir : string ) : string
10538: Function FindFirstUTF8(const Path:string; Attr: Longint; out Rslt : TSearchRec) : Longint
10539: Function FindNextUTF8( var Rslt : TSearchRec) : Longint
10540: Procedure FindCloseUTF8( var F : TSearchrec)
10541: Function FileSetDateUTF8( const FileName : String; Age : Longint ) : Longint
10542: Function FileGetAttrUTF8( const FileName : String ) : Longint
10543: Function FileSetAttrUTF8( const Filename : String; Attr : longint ) : Longint
10544: Function DeleteFileUTF8( const FileName : String ) : Boolean
10545: Function RenameFileUTF8( const OldName, NewName : String) : Boolean
10546: Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
10547: Function FileIsReadOnlyUTF8( const FileName : String ) : Boolean
10548: Function GetCurrentDirUTF8 : String
10549: Function SetCurrentDirUTF8( const NewDir : String ) : Boolean
10550: Function CreateDirUTF8( const NewDir : String ) : Boolean
10551: Function RemoveDirUTF8( const Dir : String ) : Boolean
10552: Function ForceDirectoriesUTF8( const Dir : string) : Boolean
10553: Function FileOpenUTF8( const FileName : string; Mode : Integer ) : THandle
10554: Function FileCreateUTF8( const FileName : string ) : THandle;
10555: Function FileCreateUTF81( const FileName : string; Rights : Cardinal ) : THandle;
10556: Function FileCreateUTF82( const FileName : String; ShareMode : Integer; Rights : Cardinal ) : THandle;
10557: Function FileSizeUtf8( const Filename : string ) : int64
10558: Function GetFileDescription( const AFilename : string ) : string
10559: Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
10560: Function GetAppConfigFileUTF8( Global : Boolean; SubDir : boolean; CreateDir : boolean) : string
10561: Function GetTempFileNameUTF8( const Dir, Prefix : String ) : String*)
10562: Function IsUNCPath( const Path : String ) : Boolean
10563: Function ExtractUNCVolume( const Path : String ) : String
10564: Function ExtractFileRoot( FileName : String ) : String
10565: Function GetDarwinSystemFilename( Filename : string ) : string
10566: Procedure SplitCmdLineParams( const Params : string; ParamList : TStrings; ReadBackslash : boolean)
10567: Function StrToCmdlineParam( const Param : string ) : string
10568: Function MergeCmdLineParams( ParamList : TStrings ) : string
10569: Procedure InvalidateFileStateCache( const Filename : string )
10570: Function FindAllFiles(const SearchPath:String;SearchMask:String;SearchSubDirs:Boolean):TStringList;
10571: Function FindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean ) : TStringList
10572: Function ReadFileToString( const Filename : string ) : string
10573: type
10574:   TCopyFileFlag = ( cffOverwriteFile,
10575:                      cffCreateDestDirectory, cffPreserveTime );
10576:   TCopyFileFlags = set of TCopyFileFlag;*
10577:   TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime)
10578:   TCopyFileFlags', 'set of TCopyFileFlag
10579:   Function CopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags ) : Boolean
10580: end;
10581:
10582: procedure SIRegister_lazMasks(CL: TPPascalCompiler);
10583: begin
10584:   TMaskCharType', '( mcChar, mcCharSet, mcAnyChar, mcAnyText )
10585:   SIRegister_TMask(CL);
10586:   SIRegister_TParseStringList(CL);

```

```

10587:   SIRegister_TMaskList(CL);
10588:   Function MatchesMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Boolean;
10589:   Function MatchesWindowsMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Boolean;
10590:   Function MatchesMaskList(const FileName,Mask:String;Separator:Char;const CaseSensitive:Boolean):Bool;
10591:   Function MatchesWindowsMaskList(const FileName,Mask:String;Separat:Char;const CaseSensitive:Bool):Bool;
10592: end;
10593:
10594: procedure SIRegister_JvShellHook(CL: TPSPPascalCompiler);
10595: begin
10596:   //PShellHookInfo', '^TShellHookInfo // will not work
10597:   TShellHookInfo', 'record hwnd : THandle; rc : TRect; end
10598:   SHELLHOOKINFO', 'TShellHookInfo
10599:   LPSHELLHOOKINFO', 'PShellHookInfo
10600:   TJvShellHookEvent', 'Procedure ( Sender : TObject; var Msg : TMessage)
10601:   SIRegister_TJvShellHook(CL);
10602:   Function InitJvShellHooks : Boolean
10603:   Procedure UnInitJvShellHooks
10604: end;
10605:
10606: procedure SIRegister_JvExControls(CL: TPSPPascalCompiler);
10607: begin
10608:   TDlgCode', '( dcWantAllKeys, dcWantArrows, dcWantChars, dcButton'
10609:   +', dcHasSelSel, dcWantTab, dcNative )
10610:   TDlgCodes', 'set of TDlgCode
10611:   'dcWantMessage', ' dcWantAllKeys);
10612:   SIRegister_IJvExControl(CL);
10613:   SIRegister_IJvDenySubClassing(CL);
10614:   SIRegister_TStructPtrMessage(CL);
10615:   Procedure SetDotNetBarColors( FocusedColor, UnfocusedColor : TColor)
10616:   Procedure DrawDotNetBar( Control : TWinControl; AColor : TColor; InControl : Boolean);
10617:   Procedure DrawDotNetBar( DC : HDC; R : TRect; AColor : TColor; UseFocusedColor : Boolean);
10618:   Procedure HandleDotNetBarHighlighting(Control:TWinControl;const Msg:TMessage;MouseOver:Boolean;Color:TColor);
10619:   Function CreateWMMessage( Msg : Integer; WParam : Integer; LParam : Longint) : TMessage;
10620:   Function CreateWMMessage1( Msg : Integer; WParam : Integer; LParam : TControl) : TMessage;
10621:   Function SmallPointToLong( const Pt : TSmallPoint) : Longint
10622:   Function ShiftStateToKeyData( Shift : TShiftState) : Longint
10623:   Function GetFocusedControl( AControl : TControl) : TWinControl
10624:   Function DlgcToDlgCodes( Value : Longint) : TDlgCodes
10625:   Function DlgCodesToDlgc( Value : TDlgCodes) : Longint
10626:   Procedure GetHintColor( var HintInfo : THintInfo; AControl : TControl; HintColor : TColor)
10627:   Function DispatchIsDesignMsg( Control : TControl; var Msg : TMessage) : Boolean
10628:   SIRegister_TJvExControl(CL);
10629:   SIRegister_TJvExWinControl(CL);
10630:   SIRegister_TJvExCustomControl(CL);
10631:   SIRegister_TJvExGraphicControl(CL);
10632:   SIRegister_TJvExHintWindow(CL);
10633:   SIRegister_TJvExPubGraphicControl(CL);
10634: end;
10635:
10636: (*-----*)
10637: procedure SIRegister_EncdDecd(CL: TPSPPascalCompiler);
10638: begin
10639:   Procedure EncodeStream( Input, Output : TStream)
10640:   Procedure DecodeStream( Input, Output : TStream)
10641:   Function EncodeString1( const Input : string) : string
10642:   Function DecodeString1( const Input : string) : string
10643: end;
10644:
10645: (*-----*)
10646: procedure SIRegister_SockAppReg(CL: TPSPPascalCompiler);
10647: begin
10648:   SIRegister_TWebAppRegInfo(CL);
10649:   SIRegister_TWebAppRegList(CL);
10650:   Procedure GetRegisteredWebApps( AList : TWebAppRegList)
10651:   Procedure RegisterWebApp( const AFileName, AProgID : string)
10652:   Procedure UnregisterWebApp( const AProgID : string)
10653:   Function FindRegisteredWebApp( const AProgID : string) : string
10654:   Function CreateRegistry( InitializeNewFile : Boolean) : TCustomIniFile
10655:   'sUDPPort','String 'UDPPort
10656: end;
10657:
10658: procedure SIRegister_PJEnvVars(CL: TPSPPascalCompiler);
10659: begin
10660:   // TStringDynArray', 'array of string
10661:   Function GetEnvVarValue( const VarName : string) : string
10662:   Function SetEnvVarValue( const VarName, VarValue : string) : Integer
10663:   Function DeleteEnvVar( const VarName : string) : Integer
10664:   Function CreateEnvBlock(const NewEnv:TStrings;const IncludeCurrent:Bool;const Buffer:string;const BufSize:Int):Int;
10665:   Function ExpandEnvVars( const Str : string) : string
10666:   Function GetAllEnvVars( const Vars : TStrings) : Integer
10667:   Procedure GetAllEnvVarNames( const Names : TStrings);
10668:   Function GetAllEnvVarNames1 : TStringDynArray;
10669:   Function EnvBlockSize : Integer
10670:   TPJEnvVarsEnum', 'Procedure ( const VarName : string; Data : TObject)
10671:   SIRegister_TPJEnvVarsEnumerator(CL);
10672:   SIRegister_TPJEnvVars(CL);
10673:   FindClass('TOBJECT'), 'EPJEnvVars
10674:   FindClass('TOBJECT'), 'EPJEnvVars

```

```

10675: //Procedure Register
10676: end;
10677:
10678: (*-----*)
10679: procedure SIRegister_PJConsoleApp(CL: TPPascalCompiler);
10680: begin
10681:   'cOneSecInMS', 'LongInt'( 1000 );
10682:   // 'cDefTimeSlice', 'LongInt'( 50 );
10683:   // 'cDefMaxExecTime', ' cOneMinInMS';
10684:   'cAppErrorMask', 'LongInt'( 1 shl 29 );
10685:   Function IsApplicationError( const ErrCode : LongWord ) : Boolean
10686:   TPJConsoleAppPriority', '( cpDefault, cpHigh, cpNormal, cpIdle, cpRealTime )
10687:   TPJConsoleColors', 'record Foreground : TPJConsoleColor; Background:TPJConsoleColor; end;
10688:   Function MakeConsoleColors( const AForeground, ABackground : TPJConsoleColor):TPJConsoleColors;
10689:   Function MakeConsoleColors1( const AForeground, ABackground : TColor ) : TPJConsoleColors;
10690:   Function MakeConsoleColors2( const AForeground, ABackground : TAlphaColor ) : TPJConsoleColors;
10691:   Function MakeSize( const ACX, ACY : LongInt ) : TSize
10692:   SIRegister_TPJCustomConsoleApp(CL);
10693:   SIRegister_TPJConsoleApp(CL);
10694: end;
10695:
10696: procedure SIRegister_ip_misc(CL: TPPascalCompiler);
10697: begin
10698:   INVALID_IP_ADDRESS', 'LongWord').SetUInt( $ffffffff );
10699:   t_encoding', '( uuencode, base64, mime )
10700:   Function internet_date( date : TDateTime ) : string
10701:   Function lookup_hostname( const hostname : string ) : longint
10702:   Function my_hostname : string
10703:   Function my_ip_address : longint
10704:   Function ip2string( ip_address : longint ) : string
10705:   Function resolve_hostname( ip : longint ) : string
10706:   Function address_from( const s : string; count : integer ) : string
10707:   Function encode_base64( data : TStream ) : TStringList
10708:   Function decode_base64( source : TStringList ) : TMemoryStream
10709:   Function posn( const s, t : string; count : integer ) : integer
10710:   Function poscn( c : char; const s : string; n : integer ) : integer
10711:   Function filename_of( const s : string ) : string
10712:   //Function trim( const s : string ) : string
10713:   //Procedure setlength( var s : string; l : byte )
10714:   Function TimeZoneBias : longint
10715:   Function eight2seven_quoteprint( const s : string ) : string
10716:   Function eight2seven_german( const s : string ) : string
10717:   Function seven2eight_quoteprint( const s : string ) : string end;
10718:   type in_addr', 'record s_bytes : array[1..4] of byte; end;
10719:   Function socketerror : cint
10720:   Function fpsocket( domain : cint; xtype : cint; protocol : cint ) : cint
10721:   Function fprevc( s : cint; buf : __pointer; len : size_t; flags : cint ) : ssize_t
10722:   Function fpsend( s : cint; msg : __pointer; len : size_t; flags : cint ) : ssize_t
10723:   //Function fpbind( s : cint; addrx : psockaddr; addrlen : tsocklen ) : cint
10724:   Function fplisten( s : cint; backlog : cint ) : cint
10725:   //Function fpaccept( s : cint; addrx : psockaddr; addrlen : plongint ) : cint
10726:   //Function fpconnect( s : cint; name : psockaddr; namelen : tsocklen ) : cint
10727:   //Function fpgetsockname( s : cint; name : psockaddr; namelen : psocklen ) : cint
10728:   Function NetAddrToStr( Entry : in_addr ) : String
10729:   Function HostAddrToStr( Entry : in_addr ) : String
10730:   Function StrToHostAddr( IP : String ) : in_addr
10731:   Function StrToNetAddr( IP : String ) : in_addr
10732:   SOL_SOCKET', 'LongWord').SetUInt( $ffff );
10733:   cint8', 'shortint
10734:   cuint8', 'byte
10735:   cchar', 'cint8
10736:   cschar', 'cint8
10737:   cuchar', 'cuint8
10738:   cint16', 'smallint
10739:   cuint16', 'word
10740:   cshort', 'cint16
10741:   csshort', 'cint16
10742:   cushort', 'cuint16
10743:   cint32', 'longint
10744:   cuint32', 'longword
10745:   cint', 'cint32
10746:   csint', 'cint32
10747:   cuint', 'cuint32
10748:   csigned', 'cint
10749:   cunsigned', 'cuint
10750:   cint64', 'int64
10751:   clonglong', 'cint64
10752:   cslonglong', 'cint64
10753:   cbool', 'longbool
10754:   cfloat', 'single
10755:   cdouble', 'double
10756:   clongdouble', 'extended
10757:
10758: procedure SIRegister_uLkJSON(CL: TPPascalCompiler);
10759: begin
10760:   TlkJSONTypes', '( jsBase, jsNumber, jsString, jsBoolean, jsNull, jsList, jsObject )
10761:   SIRegister_TlkJSONdotnetclass(CL);
10762:   SIRegister_TlkJSONbase(CL);
10763:   SIRegister_TlkJSONnumber(CL);

```

```

10764: SIRegister_TlkJSONstring(CL);
10765: SIRegister_TlkJSONboolean(CL);
10766: SIRegister_TlkJSONnull(CL);
10767: TlkJSONFuncEnum', 'Procedure ( ElName : string; Elem : TlkJSONba'
10768: +'se; data : TObject; var Continue : Boolean)
10769: SIRegister_TlkJSONcustomlist(CL);
10770: SIRegister_TlkJSONlist(CL);
10771: SIRegister_TlkJSONobjectmethod(CL);
10772: TlkHashItem', 'record hash : cardinal; index : Integer; end
10773: TlkHashFunction', 'Function ( const ws : WideString ) : cardinal
10774: SIRegister_TlkHashTable(CL);
10775: SIRegister_TlkBalTree(CL);
10776: SIRegister_TlkJSONobject(CL);
10777: SIRegister_TlkJSON(CL);
10778: SIRegister_TlkJSONstreamed(CL);
10779: Function GenerateReadableText( vObj : TlkJSONbase; var vLevel : Integer): string
10780: end;
10781:
10782: procedure SIRegister_ZSysUtils(CL: TPSPPascalCompiler);
10783: begin
10784:   TZListSortCompare', 'Function ( Item1, Item2 : TObject ): Integer
10785:   SIRegister_TZSortedlist(CL);
10786:   Function zFirstDelimiter( const Delimiters, Str : string ) : Integer
10787:   Function zLastDelimiter( const Delimiters, Str : string ) : Integer
10788:   //Function MemLCompUnicode( P1, P2 : PWideChar; Len : Integer ) : Boolean
10789:   //Function MemLCompAnsi( P1, P2 : PAnsiChar; Len : Integer ) : Boolean
10790:   Function zStartsWith( const Str, SubStr : WideString ) : Boolean;
10791:   Function StartsWith1( const Str, SubStr : RawByteString ) : Boolean;
10792:   Function EndsWith1( const Str, SubStr : WideString ) : Boolean;
10793:   Function EndsWith1( const Str, SubStr : RawByteString ) : Boolean;
10794:   Function SQLStrToFloatDef( Str : RawByteString; Def : Extended ) : Extended;
10795:   Function SQLStrToFloatDef1( Str : String; Def : Extended ) : Extended;
10796:   Function SQLStrToFloat( const Str : AnsiString ) : Extended
10797:   //Function BufferToStr( Buffer : PWideChar; Length : LongInt ) : string;
10798:   //Function BufferToStr1( Buffer : PAnsiChar; Length : LongInt ) : string;
10799:   Function BufferToBytes( Buffer : TObject; Length : LongInt ) : TByteDynArray
10800:   Function StrToBoolEx( Str : string ) : Boolean
10801:   Function BoolToStrEx( Bool : Boolean ) : String
10802:   Function IsIpAddr( const Str : string ) : Boolean
10803:   Function zSplitString( const Str, Delimiters : string ) : TStrings
10804:   Procedure PutSplitString( List : TStrings; const Str, Delimiters : string )
10805:   Procedure AppendSplitString( List : TStrings; const Str, Delimiters : string )
10806:   Function ComposeString( List : TStrings; const Delimiter : string ) : string
10807:   Function FloatToSQLStr( Value : Extended ) : string
10808:   Procedure PutSplitStringEx( List : TStrings; const Str, Delimiter : string )
10809:   Function SplitStringEx( const Str, Delimiter : string ) : TStrings
10810:   Procedure AppendSplitStringEx( List : TStrings; const Str, Delimiter : string )
10811:   Function zBytesToStr( const Value : TByteDynArray ) : AnsiString
10812:   Function zStrToBytes( const Value : AnsiString ) : TByteDynArray;
10813:   Function StrToBytes1( const Value : UTF8String ) : TByteDynArray;
10814:   Function StrToBytes2( const Value : RawByteString ) : TByteDynArray;
10815:   Function StrToBytes3( const Value : WideString ) : TByteDynArray;
10816:   Function StrToBytes4( const Value : UnicodeString ) : TByteDynArray;
10817:   Function BytesToVar( const Value : TByteDynArray ) : Variant
10818:   Function VarToBytes( const Value : Variant ) : TByteDynArray
10819:   Function AnsiSQLDateToDate( const Value : string ) : TDateTime
10820:   Function TimestampStrToDate( const Value : string ) : TDateTime
10821:   Function DateStrToDate( const Value : string; WithMMSec : Boolean ) : string
10822:   Function EncodeCString( const Value : string ) : string
10823:   Function DecodeCString( const Value : string ) : string
10824:   Function zReplaceChar( const Source, Target : Char; const Str : string ) : string
10825:   Function MemPas( Buffer : PChar; Length : LongInt ) : string
10826:   Procedure DecodeSQLVersioning(const FullVersion:Int;out MajorVersion:Int;out MinorVersion:Int;out
SubVersion:Int);
10827:   Function EncodeSQLVersioning(const MajorVersion:Integer;const MinorVersion:Integer;const
SubVersion:Integer):Int;
10828:   Function FormatsQLVersion( const SQLVersion : Integer ) : String
10829:   Function ZStrToFloat( Value : AnsiChar ) : Extended;
10830:   Function ZStrToFloat1( Value : AnsiString ) : Extended;
10831:   Procedure ZSetString( const Src : AnsiChar; var Dest : AnsiString );
10832:   Procedure ZSetString1( const Src : AnsiChar; const Len : Cardinal; var Dest : AnsiString );
10833:   Procedure ZSetString2( const Src : AnsiChar; var Dest : UTF8String );
10834:   Procedure ZSetString3( const Src : AnsiChar; const Len : Cardinal; var Dest : UTF8String );
10835:   Procedure ZSetString4( const Src : AnsiChar; const Len : Cardinal; var Dest : WideString );
10836:   Procedure ZSetString5( const Src : AnsiChar; var Dest : RawByteString );
10837:   Procedure ZSetString6( const Src : AnsiChar; const Len : Cardinal; var Dest : RawByteString );
10838: end;
10839:
10840: unit uPSI_ZEncoding;
10841:   Function StringToAnsiEx( const s : String; const FromCP, ToCP : Word ) : RawByteString
10842:   Function AnsiToStringEx( const s : RawByteString; const FromCP, ToCP : Word ) : String
10843:   Function ZRawToUnicode( const S : RawByteString; const CP : Word ) : WideString
10844:   Function ZUnicodeToRaw( const US : WideString; CP : Word ) : RawByteString
10845:   Function ZConvertAnsiToRaw( const Src : AnsiString; const RawCP : Word ) : RawByteString
10846:   Function ZConvertRawToAnsi( const Src : RawByteString; const RawCP : Word ) : AnsiString
10847:   Function ZConvertAnsiToUTF8( const Src : AnsiString ) : UTF8String
10848:   Function ZConvertUTF8ToAnsi( const Src : UTF8String ) : AnsiString
10849:   Function ZConvertRawToUTF8( const Src : RawByteString; const CP : Word ) : UTF8String
10850:   Function ZConvertUTF8ToRaw( const Src : UTF8String; const CP : Word ) : RawByteString

```

```

10851: Function ZConvertRawToString( const Src : RawByteString; const RawCP, StringCP : Word ) : String
10852: Function ZConvertStringToRaw( const Src : String; const StringCP, RawCP : Word ) : RawByteString
10853: Function ZConvertStringToRawWithAutoEncode(const Src:String;const StringCP,RawCP:Word):RawByteString;
10854: Function ZConvertUTF8ToString( const Src : UTF8String; const StringCP : Word ) : String
10855: Function ZConvertStringToUTF8( const Src : String; const StringCP : Word ) : UTF8String
10856: Function ZConvertStringToUTF8WithAutoEncode( const Src : String; const StringCP: Word) : UTF8String
10857: Function ZConvertStringToAnsi( const Src : String; const StringCP : Word ) : AnsiString
10858: Function ZConvertStringToAnsiWithAutoEncode( const Src : String; const StringCP: Word): AnsiString
10859: Function ZConvertAnsiToString( const Src : AnsiString; const StringCP : Word ) : String
10860: Function ZConvertUnicodeToString( const Src : WideString; const StringCP : Word ) : String
10861: Function ZConvertUnicodeToString_CPUTF8( const Src : WideString; const StringCP : Word ) : String
10862: Function ZConvertStringToUnicode( const Src : String; const StringCP : Word ) : WideString
10863: Function ZConvertString_CPUTF8ToUnicode( const Src : String; const StringCP : Word ) : WideString
10864: Function ZConvertStringToUnicodeWithAutoEncode( const Src: String;const StringCP:Word):WideString
10865: Function ZMoveAnsiToRaw( const Src : AnsiString; const RawCP : Word ) : RawByteString
10866: Function ZMoveRawToAnsi( const Src : RawByteString; const RawCP : Word ) : AnsiString
10867: Function ZMoveAnsiToUTF8( const Src : AnsiString ) : UTF8String
10868: Function ZMoveUTF8ToAnsi( const Src : UTF8String ) : AnsiString
10869: Function ZMoveRawToUTF8( const Src : RawByteString; const CP : Word ) : UTF8String
10870: Function ZMoveUTF8ToRaw( const Src : UTF8String; const CP : Word ) : RawByteString
10871: Function ZMoveStringToAnsi( const Src : String; const StringCP : Word ) : AnsiString
10872: Function ZMoveAnsiToString( const Src : AnsiString; const StringCP : Word ) : String
10873: Function ZMoveRawToString( const Src : RawByteString; const RawCP, StringCP : Word ) : String
10874: Function ZMoveStringToRaw( const Src : String; const StringCP, RawCP : Word ) : RawByteString
10875: Function ZMoveUTF8ToString( const Src : UTF8String; StringCP : Word ) : String
10876: Function ZMoveStringToUTF8( const Src : String; const StringCP : Word ) : UTF8String
10877: Function ZUnknownRawToUnicode( const S : RawByteString; const CP : Word ) : WideString
10878: Function ZUnknownRawToUnicodeWithAutoEncode( const S : RawByteString; const CP : Word ) : WideString
10879: Function ZUnicodeToUnknownRaw( const US : WideString; CP : Word ) : RawByteString
10880: Function ZDefaultSystemCodePage : Word
10881: Function ZCompatibleCodePages( const CP1, CP2 : Word ) : Boolean
10882:
10883:
10884: procedure SIRegister_BoldComUtils(CL: TPSPPascalCompiler);
10885: begin
10886:   ('RPC_C_AUTHN_LEVEL_DEFAULT','LongInt'( 0 );
10887:   ('RPC_C_AUTHN_LEVEL_NONE','LongInt'( 1 );
10888:   ('RPC_C_AUTHN_LEVEL_CONNECT','LongInt'( 2 );
10889:   ('RPC_C_AUTHN_LEVEL_CALL','LongInt'( 3 );
10890:   ('RPC_C_AUTHN_LEVEL_PKT','LongInt'( 4 );
10891:   ('RPC_C_AUTHN_LEVEL_PKT_INTEGRITY','LongInt'( 5 );
10892:   ('RPC_C_AUTHN_LEVEL_PKT_PRIVACY','LongInt'( 6 );
10893:   {'alDefault', '1 RPC_C_AUTHN_LEVEL_DEFAULT';
10894:   ('alNone', '2 RPC_C_AUTHN_LEVEL_NONE';
10895:   ('alConnect', '3 RPC_C_AUTHN_LEVEL_CONNECT';
10896:   ('alCall', '4 RPC_C_AUTHN_LEVEL_CALL;
10897:   ('alPacket', '5 RPC_C_AUTHN_LEVEL_PKT;
10898:   ('alPacketIntegrity', '6 RPC_C_AUTHN_LEVEL_PKT_INTEGRITY;
10899:   ('alPacketPrivacy', '7 RPC_C_AUTHN_LEVEL_PKT_PRIVACY );
10900:   ('RPC_C_IMP_LEVEL_DEFAULT','LongInt'( 0 );
10901:   ('RPC_C_IMP_LEVEL_ANONYMOUS','LongInt'( 1 );
10902:   ('RPC_C_IMP_LEVEL_IDENTIFY','LongInt'( 2 );
10903:   ('RPC_C_IMP_LEVEL_IMPERSONATE','LongInt'( 3 );
10904:   ('RPC_C_IMP_LEVEL_DELEGATE','LongInt'( 4 );
10905:   {'ilDefault', '0 RPC_C_IMP_LEVEL_DEFAULT';
10906:   ('ilAnonymous', '1 RPC_C_IMP_LEVEL_ANONYMOUS;
10907:   ('ilIdentiry', '2 RPC_C_IMP_LEVEL_IDENTIFY;
10908:   ('ilImpersonate', '3 RPC_C_IMP_LEVEL_IMPERSONATE;
10909:   ('ilDelegate', '4 RPC_C_IMP_LEVEL_DELEGATE );
10910:   ('EOAC_NONE','LongWord').SetUInt( $0 );
10911:   ('EOAC_DEFAULT','LongWord').SetUInt( $800 );
10912:   ('EOAC_MUTUAL_AUTH','LongWord').SetUInt( $1 );
10913:   ('EOAC_STATIC_CLOACKING','LongWord').SetUInt( $20 );
10914:   ('EOAC_DYNAMIC_CLOAKING','LongWord').SetUInt( $40 );
10915:   ('EOAC_ANY_AUTHORITY','LongWord').SetUInt( $80 );
10916:   ('RPC_C_AUTHN_WINNT','LongInt'( 10 );
10917:   ('RPC_C_AUTHNZ_NONE','LongInt'( 0 );
10918:   ('RPC_C_AUTHNZ_NAME','LongInt'( 1 );
10919:   ('RPC_C_AUTHNZ_DCE','LongInt'( 2 );
10920:   FindClass('TObject'), 'EBoldCom
10921: Function BoldVariantIsType( V : OleVariant; TypeCode : Integer ) : Boolean
10922: Function BoldMemoryToVariant( const Buffer, BufSize : Integer ) : OleVariant
10923: Function BoldStreamToVariant( Stream : TStream ) : OleVariant
10924: Function BoldStringsToVariant( Strings : TStrings ) : OleVariant
10925: Function BoldVariantToMemory( V : OleVariant; var Buffer, BufSize : Integer ) : Integer
10926: Function BoldVariantToStream( V : OleVariant; Stream : TStream ) : Integer
10927: Function BoldVariantArrayOfArraysOfStringToStrings( V : OleVariant; Strings : TStrings ) : Integer
10928: Function BoldVariantIsNamedValues( V : OleVariant ) : Boolean
10929: Function BoldCreateNamedValues(const Names:array of string;const Values:array of OleVariant):OleVariant;
10930: Function BoldGetNamedValue( Data : OleVariant; const Name : string ) : OleVariant
10931: Procedure BoldSetNamedValue( Data : OleVariant; const Name : string; Value : OleVariant )
10932: Function BoldCreateGUID : TGUID
10933: Function BoldCreateComObject( const ClsId, IID : TGUID; out Obj : variant; out Res : HResult ) : Boolean
10934: Function BoldCreateRemoteComObject(const HostName:string;const ClsId,IId:TGUID;out Obj:variant;out Res:HRes):Bool;
10935: Procedure BoldInitializeComSecurity( AuthenticationLevel, ImpersonationLevel : longint )
10936: Procedure BoldSetSecurityForInterface(AuthenticationLevel,ImpersonationLevel:longint;Unk:IUnknown );
10937: end;
10938:

```

```

10939: (*-----*)
10940: procedure SIRegister_BoldIsoDateTime(CL: TPSPascalCompiler);
10941: begin
10942:   Function ParseISODate( s : string ) : TDateTime
10943:   Function ParseISODateTime( s : string ) : TDateTime
10944:   Function ParseISOTime( str : string ) : TDateTime
10945: end;
10946:
10947: (*-----*)
10948: procedure SIRegister_BoldGUIDUtils(CL: TPSPascalCompiler);
10949: begin
10950:   Function BoldCreateGUIDAsString( StripBrackets : Boolean ) : string
10951:   Function BoldCreateGUIDwithBracketsAsString : string
10952: end;
10953:
10954: procedure SIRegister_BoldFileHandler(CL: TPSPascalCompiler);
10955: begin
10956:   FindClass('TOBJECT'), 'TBoldFileHandler'
10957:   FindClass('TOBJECT'), 'TBoldDiskFileHandler'
10958:   //TBoldFileHandlerClass', 'class of TBoldFileHandler
10959:   TBoldInitializeFileContents', 'Procedure ( StringList : TStringList )
10960:   SIRegister_TBoldFileHandler(CL);
10961:   SIRegister_TBoldDiskFileHandler(CL);
10962:   Procedure BoldCloseAllFilehandlers
10963:   Procedure BoldRemoveUnchangedFilesFromEditor
10964:   Function BoldFileHandlerList : TBoldObjectArray
10965:   Function BoldFileHandlerForfile(path,FileName:String; ModuleType:TBoldModuleType;ShowInEditor:Bool;
10966:   OnInitializeFileContents : TBoldInitializeFileContents) : TBoldFileHandler
10967: end;
10968: procedure SIRegister_BoldWinINet(CL: TPSPascalCompiler);
10969: begin
10970:   PCharArr', 'array of PChar
10971:   Function BoldInternetOpen(Agent:String;
10972:   AccessType:integer;Proxy:string;ProxyByPass:String;Flags:integer):ptr;
10973:   Function BoldInternetOpenUrl(iNet:Pointer;URL: string; Headers:String;Flags,Context:cardinal):Pointer
10974:   Function BoldInternetReadFile(hFile:Pointer;Buff:Ptr;NumbOfBytesToRead:Card;var
10975:   NumberOfBytesRead:Card):LongBool;
10976:   Function BoldInternetCloseHandle( HINet : Pointer ) : LongBool
10977:   Function BoldHttpQueryInfo( hRequest : Pointer; InfoLevel : Cardinal; Buffer : Pointer; BufferLength :
10978:   Cardinal; Reserved : Cardinal ) : LongBool
10979:   Function BoldInternetQueryDataAvailable( hFile : Pointer; var NumberOfBytesAvailable : Cardinal; flags :
10980:   Cardinal; Context : Cardinal ) : LongBool
10981:   Function BoldHttpOpenRequest(hConnect: Pointer; Verb, ObjectName, Version, Referrer : String; AcceptTypes
10982:   : PCharArr; Flags, Context : Cardinal) : Pointer
10983:   Function BoldHttpSendRequest(hRequest:Ptr;Headers:string;Optional:Ptr;OptionalLength:Cardinal): LongBool
10984:   Function BoldInternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,DWORD;var lppvData:Ptr):DWORD
10985:   Function BoldInternetAttemptConnect( dwReserved : DWORD ) : DWORD
10986:   Function BoldInternetConnect(hInet: HINTERNET;ServerName:string; nServerPort:INTERNET_PORT;
10987:   Username:string; Password : string; dwService : DWORD; dwFlags : DWORD; dwContext : DWORD):HINTERNET
10988:   Function BoldInternetCrackUrl(Url:PChar;UrlLength,dwFlags:DWORD;var lpUrlComponents:TURLComponents):BOOL;
10989: end;
10990:
10991:
10992: (*-----*)
10993: procedure SIRegister_BoldQueue(CL: TPSPascalCompiler);
10994: begin
10995:   //('befIsInDisplayList',' BoldElementFlag0 );
10996:   //('befStronglyDependedOfPrioritized',' BoldElementFlag1 );
10997:   //('befFollowerSelected',' BoldElementFlag2 );
10998:   FindClass('TOBJECT'), 'TBoldQueue
10999:   FindClass('TOBJECT'), 'TBoldQueueable
11000:   TBoldQueueDisplayMode', '( dmDisplayOne, dmDisplayAll )
11001:   SIRegister_TBoldQueueable(CL);
11002:   SIRegister_TBoldQueue(CL);
11003:   Function BoldQueueFinalized : Boolean
11004:   Function BoldInstalledQueue : TBoldQueue
11005: end;
11006:
11007: procedure SIRegister_BarcodE(CL: TPSPascalCompiler);
11008: begin
11009:   const mmPerInch', 'Extended').setExtended( 25.4);
11010:   TBarcodeType', '( bcCode_2_5_interleaved, bcCode_2_5_industrial,'
11011:   +' bcCode_2_5_matrix, bcCode39, bcCode39Extended, bcCode128A, bcCode128B, bc'
11012:   +' Code128C, bcCode93, bcCode93Extended, bcCodeMSI, bcCodePostNet, bcCodeCoda'
11013:   +' bar, bcCodeEAN8, bcCodeEAN13, bcCodeUPC_A, bcCodeUPC_E0, bcCodeUPC_E1, bcC'
11014:   +' odeUPC_Supp2, bcCodeUPC_Supp5, bcCodeEAN128A, bcCodeEAN128B, bcCodeEAN128C
11015:   TBarLineType', '( white, black, black_half )
11016:   TBarcodeOption', '( bcoNone, bcoCode, bcoTyp, bcoBoth )
11017:   TShowTextPosition', '( stpTopLeft, stpTopRight, stpTopCenter, st'
11018:   +' pBottomLeft, stpBottomRight, stpBottomCenter )
11019:   TCheckSumMethod', '( csmNone, csmModulo10 )
11020:   SIRegister_TAsBarcode(CL);

```

```

11021: Function CheckSumModulo10( const data : string ) : string
11022: Function ConvertMmToPixelsX( const Value : Double ) : Integer
11023: Function ConvertMmToPixelsY( const Value : Double ) : Integer
11024: Function ConvertInchToPixelsX( const Value : Double ) : Integer
11025: Function ConvertInchToPixelsY( const Value : Double ) : Integer
11026: end;
11027:
11028: procedure SIRegister_Geometry(CL: TPSPascalCompiler); //OpenGL
11029: begin
11030:   THomogeneousByteVector', 'array[0..3] of Byte
11031:   THomogeneousWordVector', 'array[0..3] of Word
11032:   THomogeneousIntVector', 'array[0..3] of Integer
11033:   THomogeneousFltVector', 'array[0..3] of single
11034:   THomogeneousDblVector', 'array[0..3] of double
11035:   THomogeneousExtVector', 'array[0..3] of extended
11036:   TAffineByteVector', 'array[0..2] of Byte
11037:   TAffineWordVector', 'array[0..2] of Word
11038:   TAffineIntVector', 'array[0..2] of Integer
11039:   TAffineFltVector', 'array[0..2] of single
11040:   TAffineDblVector', 'array[0..2] of double
11041:   TAffineExtVector', 'array[0..2] of extended
11042:   THomogeneousByteMatrix', 'array[0..3] of THomogeneousByteVector
11043:   THomogeneousWordMatrix', 'array[0..3] of THomogeneousWordVector
11044:   THomogeneousIntMatrix', 'array[0..3] of THomogeneousIntVector
11045:   THomogeneousFltMatrix', 'array[0..3] of THomogeneousFltVector
11046:   THomogeneousDblMatrix', 'array[0..3] of THomogeneousDblVector
11047:   THomogeneousExtMatrix', 'array[0..3] of THomogeneousExtVector
11048:   TAffineByteMatrix', 'array[0..2] of TAffineByteVector
11049:   TAffineWordMatrix', 'array[0..2] of TAffineWordVector
11050:   TAffineIntMatrix', 'array[0..2] of TAffineIntVector
11051:   TAffineFltMatrix', 'array[0..3] of TAffineFltVector
11052:   TAffineDblMatrix', 'array[0..3] of TAffineDblVector
11053:   TAffineExtMatrix', 'array[0..3] of TAffineExtVector
11054:   TMatrix4b', 'THomogeneousByteMatrix
11055:   TMatrix4w', 'THomogeneousWordMatrix
11056:   TMatrix4i', 'THomogeneousIntMatrix
11057:   TMatrix4f', 'THomogeneousFltMatrix
11058:   TMatrix4d', 'THomogeneousDblMatrix
11059:   TMatrix4e', 'THomogeneousExtMatrix
11060:   TMatrix3b', 'TAffineByteMatrix
11061:   TMatrix3w', 'TAffineWordMatrix
11062:   TMatrix3i', 'TAffineIntMatrix
11063:   TMatrix3f', 'TAffineFltMatrix
11064:   TMatrix3d', 'TAffineDblMatrix
11065:   TMatrix3e', 'TAffineExtMatrix
11066: //'PMatrix', '^TMatrix // will not work
11067: TMatrixGL', 'THomogeneousFltMatrix
11068: THomogeneousMatrix', 'THomogeneousFltMatrix
11069: TAffineMatrix', 'TAffineFltMatrix
11070: TQuaternion', 'record Vector : TVector4f; end
11071: TRectangle', 'record Left : integer; Top : integer; Width : integer;
11072: +'ger; Height : Integer; end
11073: TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
11074:   +'XZ, ttShearyYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
11075:   +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )
11076: 'EPSILON', 'Extended').setExtended( 1E-100 );
11077: 'EPSILON2', 'Extended').setExtended( 1E-50 );
11078: Function VectorAddGL( V1, V2 : TVectorGL ) : TVectorGL
11079: Function VectorAffineAdd( V1, V2 : TAffineVector ) : TAffineVector
11080: Function VectorAffineCombine(V1,V2:TAffineVector; F1, F2 : Single) : TAffineVector
11081: Function VectorAffineDotProduct( V1, V2 : TAffineVector ) : Single
11082: Function VectorAffineLerp( V1, V2 : TAffineVector; t : Single ) : TAffineVector
11083: Function VectorAffineSubtract( V1, V2 : TAffineVector ) : TAffineVector
11084: Function VectorAngle( V1, V2 : TAffineVector ) : Single
11085: Function VectorCombine( V1, V2 : TVectorGL; F1, F2 : Single ) : TVectorGL
11086: Function VectorCrossProduct( V1, V2 : TAffineVector ) : TAffineVector
11087: Function VectorDotProduct( V1, V2 : TVectorGL ) : Single
11088: Function VectorLength( V : array of Single ) : Single
11089: Function VectorLerp( V1, V2 : TVectorGL; t : Single ) : TVectorGL
11090: Procedure VectorNegate( V : array of Single )
11091: Function VectorNorm( V : array of Single ) : Single
11092: Function VectorNormalize( V : array of Single ) : Single
11093: Function VectorPerpendicular( V, N : TAffineVector ) : TAffineVector
11094: Function VectorReflect( V, N : TAffineVector ) : TAffineVector
11095: Procedure VectorRotate( var Vector : TVector4f; Axis : TVector3f; Angle : Single )
11096: Procedure VectorScale( V : array of Single; Factor : Single )
11097: Function VectorSubtractGL( V1, V2 : TVectorGL ) : TVectorGL
11098: Function CreateRotationMatrixX( Sine, Cosine : Single ) : TMatrixGL
11099: Function CreateRotationMatrixY( Sine, Cosine : Single ) : TMatrixGL
11100: Function CreateRotationMatrixZ( Sine, Cosine : Single ) : TMatrixGL
11101: Function CreateScaleMatrix( V : TAffineVector ) : TMatrixGL
11102: Function CreateTranslationMatrix( V : TVectorGL ) : TMatrixGL
11103: Procedure MatrixAdjoint( var M : TMatrixGL )
11104: Function MatrixAffineDeterminant( M : TAffineMatrix ) : Single
11105: Procedure MatrixAffineTranspose( var M : TAffineMatrix )
11106: Function MatrixDeterminant( M : TMatrixGL ) : Single
11107: Procedure MatrixInvert( var M : TMatrixGL )
11108: Function MatrixMultiply( M1, M2 : TMatrixGL ) : TMatrixGL
11109: Procedure MatrixScale( var M : TMatrixGL; Factor : Single )

```

```

11110: Procedure MatrixTranspose( var M : TMatrixGL)
11111: Function QuaternionConjugate( Q : TQuaternion ) : TQuaternion
11112: Function QuaternionFromPoints( V1, V2 : TAffineVector ) : TQuaternion
11113: Function QuaternionMultiply( qL, qR : TQuaternion ) : TQuaternion
11114: Function QuaternionSlerp( QStart,QEnd:TQuaternion; Spin:Integer; t:Single):TQuaternion
11115: Function QuaternionToMatrix( Q : TQuaternion ) : TMatrixGL
11116: Procedure QuaternionToPoints( Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector )
11117: Function ConvertRotation( Angles : TAffineVector ) : TVectorGL
11118: Function CreateRotationMatrix( Axis : TVector3f; Angle : Single ) : TMatrixGL
11119: //Function MatrixDecompose( M : TMatrixGL; var Tran : TTransformations ) : Boolean
11120: Function VectorAffineTransform( V : TAffineVector; M : TAffineMatrix ) : TAffineVector
11121: Function VectorTransform( V : TVector4f; M : TMatrixGL ) : TVector4f;
11122: Function VectorTransform1( V : TVector3f; M : TMatrixGL ) : TVector3f;
11123: Function MakeAffineDblVector( V : array of Double ) : TAffineDblVector
11124: Function MakeDblVector( V : array of Double ) : THomogeneousDblVector
11125: Function MakeAffineVector( V : array of Single ) : TAffineVector
11126: Function MakeQuaternion( Imag : array of Single; Real : Single ) : TQuaternion
11127: Function MakeVector( V : array of Single ) : TVectorGL
11128: Function PointInPolygonGL( xp, yp : array of Single; x, y : Single ) : Boolean
11129: Function VectorAffineDblToFlt( V : TAffineDblVector ) : TAffineVector
11130: Function VectorDblToFlt( V : THomogeneousDblVector ) : THomogeneousVector
11131: Function VectorAffineFltToDbl( V : TAffineVector ) : TAffineDblVector
11132: Function VectorFltToDbl( V : TVectorGL ) : THomogeneousDblVector
11133: Function ArcCosGL( X : Extended ) : Extended
11134: Function ArcSInGL( X : Extended ) : Extended
11135: Function ArcTan2GL( Y, X : Extended ) : Extended
11136: Function CoTanGL( X : Extended ) : Extended
11137: Function DegToRadGL( Degrees : Extended ) : Extended
11138: Function RadToDegGL( Radians : Extended ) : Extended
11139: Procedure SinCosGL( Theta : Extended; var Sin, Cos : Extended )
11140: Function TanGL( X : Extended ) : Extended
11141: Function Turn( Matrix : TMatrixGL; Angle : Single ) : TMatrixGL;
11142: Function Turn1( Matrix : TMatrixGL; MasterUp : TAffineVector; Angle: Single): TMatrixGL;
11143: Function Pitch( Matrix : TMatrixGL; Angle : Single ) : TMatrixGL;
11144: Function Pitch1( Matrix : TMatrixGL; MasterRight:TAffineVector;Angle:Single):TMatrixGL;
11145: Function Roll( Matrix : TMatrixGL; Angle : Single ) : TMatrixGL;
11146: Function Roll1( Matrix:TMatrixGL; MasterDirection:TAffineVector;Angle:Single):TMatrixGL;
11147: end;
11148:
11149:
11150: procedure SIRegister_JclRegistry(CL: TPSPascalCompiler);
11151: begin
11152:   Function RegCreateKey( const RootKey : HKEY; const Key, Value : string ) : Longint
11153:   Function RegDeleteEntry( const RootKey : HKEY; const Key, Name : string ) : Boolean
11154:   Function RegDeleteKeyTree( const RootKey : HKEY; const Key : string ) : Boolean
11155:   Function RegReadBool( const RootKey : HKEY; const Key, Name : string ) : Boolean
11156:   Function RegReadBoolDef( const RootKey : HKEY; const Key, Name : string; Def : Boolean ) : Boolean
11157:   Function RegReadInteger( const RootKey : HKEY; const Key, Name : string ) : Integer
11158:   Function RegReadIntegerDef( const RootKey : HKEY; const Key, Name : string; Def : Integer ) : Integer
11159:   Function RegReadString( const RootKey : HKEY; const Key, Name : string ) : string
11160:   Function RegReadStringDef( const RootKey : HKEY; const Key, Name, Def : string ) : string
11161:   Function RegReadDWORD( const RootKey : HKEY; const Key, Name : string ) : Int64
11162:   Function RegReadDWORDDef( const RootKey : HKEY; const Key, Name : string; Def : Int64 ) : Int64
11163:   Procedure RegWriteBool( const RootKey : HKEY; const Key, Name : string; Value : Boolean )
11164:   Procedure RegWriteInteger( const RootKey : HKEY; const Key, Name : string; Value : Integer )
11165:   Procedure RegWriteString( const RootKey : HKEY; const Key, Name, Value : string )
11166:   Procedure RegWriteDWORD( const RootKey : HKEY; const Key, Name : string; Value : Int64 )
11167:   Function RegGetValueNames( const RootKey : HKEY; const Key : string; const List : TStrings ) : Boolean
11168:   Function RegGetKeyNames( const RootKey : HKEY; const Key : string; const List : TStrings ) : Boolean
11169:   Function RegHasSubKeys( const RootKey : HKEY; const Key : string ) : Boolean
11170:   Function RegKeyExists( const RootKey : HKEY; const Key : string ) : Boolean
11171:   AddTypeS('TExecKind', '( ekMachineRun, ekMachineRunOnce, ekUserRun, ekUser'
11172:             + 'RunOnce, ekServiceRun, ekServiceRunOnce )'
11173:             + 'AddClassN(FindClass(''TOBJECT''), ''JclRegistryError'
11174:             + 'Function UnregisterAutoExec( ExecKind : TExecKind; const Name : string ) : Boolean
11175:             + 'Function RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline : string ) : Boolean
11176:             + 'Function RegSaveList(const RootKey:HKEY;const Key:string; const ListName:string;const
11177:             Items:TStrings):Bool;
11178:             + 'Function RegLoadList(const RootKey:HKEY;const Key:string;const ListName:string;const
11179:             SaveTo:TStrings):Bool;
11180:             + 'Function RegDelList( const RootKey:HKEY;const Key:string; const ListName:string): Boolean
11181:   end;
11182: procedure SIRegister_JclCOM(CL: TPSPascalCompiler);
11183: begin
11184:   CLSID_StdComponentCategoriesMgr', 'GUID '{0002E005-0000-0000-C000-000000000046}
11185:   CATID_SafeForInitializing', 'GUID '{7DD95802-9882-11CF-9FA9-00AA006C42C4}
11186:   CATID_SafeForScripting', 'GUID '{7DD95801-9882-11CF-9FA9-00AA006C42C4}
11187:   icMAX_CATEGORY_DESC_LEN', 'LongInt'( 128 );
11188:   FindClass('TOBJECT'), 'EInvalidParam
11189:   Function IsDCOMInstalled : Boolean
11190:   Function IsDCOMEnabled : Boolean
11191:   Function GetDCOMVersion : string
11192:   Function GetMDACVersion : string
11193:   Function GetMDACVersion2 : string
11194:   Function MarshalInterThreadInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
11195:   VarArray:OleVariant):HResult;
11196:   Function MarshalInterProcessInterfaceInStream(const iid:TIID;unk:IUnknown;var
11197:   VarArray:OleVariant):HResult;

```

```

11196: Function MarshalInterMachineInterfaceInStream( const iid:TIID;unk:IUnknown;var stm:IStream):HRESULT;
11197: Function MarshalInterMachineInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
VarArray:OleVariant):HRESULT;
11198: Function CreateComponentCategory( const CatID : TGUID; const sDescription : string ) : HRESULT
11199: Function RegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID ) : HRESULT
11200: Function UnRegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID ) : HRESULT
11201: Function ResetIStreamToStart( Stream : IStream ) : Boolean
11202: Function SizeOfIStreamContents( Stream : IStream ) : Largeint
11203: Function StreamToVariantArray( Stream : TStream ) : OleVariant;
11204: Function StreamToVariantArray1( Stream : IStream ) : OleVariant;
11205: Procedure VariantArrayToStream( VarArray : OleVariant; var Stream : TStream );
11206: Procedure VariantArrayToStream1( VarArray : OleVariant; var Stream : IStream );
11207: end;
11208:
11209:
11210: procedure SIRegister_JclUnitConv_mX2(CL: TPSPascalCompiler);
11211: begin
11212:   Const ('CelsiusFreezingPoint','Extended').setExtended( 0.0 );
11213:   FahrenheitFreezingPoint,'Extended').setExtended( 32.0 );
11214:   KelvinFreezingPoint,'Extended').setExtended( 273.15 );
11215:   CelsiusAbsoluteZero','Extended').setExtended( - 273.15 );
11216:   FahrenheitAbsoluteZero','Extended').setExtended( - 459.67 );
11217:   KelvinAbsoluteZero','Extended').setExtended( 0.0 );
11218:   DegPerCycle ','Extended').setExtended( 360.0 );
11219:   DegPerGrad ','Extended').setExtended( 0.9 );
11220:   DegPerRad ','Extended').setExtended( 57.295779513082320876798154814105 );
11221:   GradPerCycle ','Extended').setExtended( 400.0 );
11222:   GradPerDeg ','Extended').setExtended( 1.111111111111111111111111111111 );
11223:   GradPerRad ','Extended').setExtended( 63.661977236758134307553505349006 );
11224:   RadPerCycle ','Extended').setExtended( 6.283185307179586476925286766559 );
11225:   RadPerDeg ','Extended').setExtended( 0.017453292519943295769236907684886 );
11226:   RadPerGrad ','Extended').setExtended( 0.015707963267948966192313216916398 );
11227:   CyclePerDeg ','Extended').setExtended( 0.00277777777777777777777777777777 );
11228:   CyclePerGrad ','Extended').setExtended( 0.0025 );
11229:   CyclePerRad ','Extended').setExtended( 0.15915494309189533576888376337251 );
11230:   ArcMinutesPerDeg ','Extended').setExtended( 60.0 );
11231:   ArcSecondsPerArcMinute ','Extended').setExtended( 60.0 );
11232:   Function HowAOneLinerCanBiteYou( const Step, Max : Longint ) : Longint
11233:   Function MakePercentage( const Step, Max : Longint ) : Longint
11234:   Function CelsiusToKelvin( const T : double ) : double
11235:   Function CelsiusToFahrenheit( const T : double ) : double
11236:   Function KelvinToCelsius( const T : double ) : double
11237:   Function KelvinToFahrenheit( const T : double ) : double
11238:   Function FahrenheitToCelsius( const T : double ) : double
11239:   Function FahrenheitToKelvin( const T : double ) : double
11240:   Function CycleToDeg( const Cycles : double ) : double
11241:   Function CycleToGrad( const Cycles : double ) : double
11242:   Function CycleToRad( const Cycles : double ) : double
11243:   Function DegToCycle( const Degrees : double ) : double
11244:   Function DegToGrad( const Degrees : double ) : double
11245:   Function DegToRad( const Degrees : double ) : double
11246:   Function GradToCycle( const Grads : double ) : double
11247:   Function GradToDeg( const Grads : double ) : double
11248:   Function GradToRad( const Grads : double ) : double
11249:   Function RadToCycle( const Radians : double ) : double
11250:   Function RadToDeg( const Radians : double ) : double
11251:   Function RadToGrad( const Radians : double ) : double
11252:   Function DmsToDeg( const D, M : Integer; const S : double ) : double
11253:   Function DmsToRad( const D, M : Integer; const S : double ) : double
11254:   Procedure DegToDms( const Degrees : double; out D, M : Integer; out S : double )
11255:   Function DegToDmsStr( const Degrees : double; const SecondPrecision : Cardinal ) : string
11256:   Procedure CartesianToPolar( const X, Y : double; out R, Phi : double )
11257:   Procedure PolarToCartesian( const R, Phi : double; out X, Y : double )
11258:   Procedure CartesianToCylinder( const X, Y, Z : double; out R, Phi, Zeta : double )
11259:   Procedure CartesianToSpheric( const X, Y, Z : double; out Rho, Phi, Theta : double )
11260:   Procedure CylinderToCartesian( const R, Phi, Zeta : double; out X, Y, Z : double )
11261:   Procedure SphericToCartesian( const Rho, Theta, Phi : double; out X, Y, Z : double )
11262:   Function CmToInch( const Cm : double ) : double
11263:   Function InchToCm( const Inch : double ) : double
11264:   Function FeetToMetre( const Feet : double ) : double
11265:   Function MetreToFeet( const Metre : double ) : double
11266:   Function YardToMetre( const Yard : double ) : double
11267:   Function MetreToYard( const Metre : double ) : double
11268:   Function NmToKm( const Nm : double ) : double
11269:   Function KmToNm( const Km : double ) : double
11270:   Function KmToSm( const Km : double ) : double
11271:   Function SmToKm( const Sm : double ) : double
11272:   Function LitreToGalUs( const Litre : double ) : double
11273:   Function GalUsToLitrel( const GalUs : double ) : double
11274:   Function GalUsToGalCan( const GalUs : double ) : double
11275:   Function GalCanToGalUs( const GalCan : double ) : double
11276:   Function GalUsToGalUk( const GalUs : double ) : double
11277:   Function GalUkToGalUs( const GalUk : double ) : double
11278:   Function LitreToGalCan( const Litre : double ) : double
11279:   Function GalCanToLitre( const GalCan : double ) : double
11280:   Function LitreToGalUk( const Litre : double ) : double
11281:   Function GalUkToLitrel( const GalUk : double ) : double
11282:   Function KgToLb( const Kg : double ) : double
11283:   Function LbToKg( const Lb : double ) : double

```

```

11284: Function KgToOz( const Kg : double) : double
11285: Function OzToKg( const Oz : double) : double
11286: Function CwtUsToKg( const Cwt : double) : double
11287: Function CwtUkToKg( const Cwt : double) : double
11288: Function KaratToKg( const Karat : double) : double
11289: Function KgToCwtUs( const Kg : double) : double
11290: Function KgToCwtUk( const Kg : double) : double
11291: Function KgToKarat( const Kg : double) : double
11292: Function KgToSton( const Kg : double) : double
11293: Function KgToLton( const Kg : double) : double
11294: Function StonToKg( const STon : double) : double
11295: Function LtonToKg( const Lton : double) : double
11296: Function QrUsToKg( const Qr : double) : double
11297: Function QrUkToKg( const Qr : double) : double
11298: Function KgToQrUs( const Kg : double) : double
11299: Function KgToQrUk( const Kg : double) : double
11300: Function PascalToBar( const Pa : double) : double
11301: Function PascalToAt( const Pa : double) : double
11302: Function PascalToTorr( const Pa : double) : double
11303: Function BarToPascal( const Bar : double) : double
11304: Function AtToPascal( const At : double) : double
11305: Function TorrToPascal( const Torr : double) : double
11306: Function KnotToMs( const Knot : double) : double
11307: Function HpElectricToWatt( const HpE : double) : double
11308: Function HpMetricToWatt( const HpM : double) : double
11309: Function MsToKnot( const ms : double) : double
11310: Function WattToHpElectric( const W : double) : double
11311: Function WattToHpMetric( const W : double) : double
11312: function getBigPI: string; //PI of 1000 numbers
11313:
11314: procedure SIRegister_devutils(CL: TPSPPascalCompiler);
11315: begin
11316:   Function CDEExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
11317:   Procedure CDCopyFile( const FileName, DestName : string)
11318:   Procedure CDMoveFile( const FileName, DestName : string)
11319:   Function MakeCommaTextToColor( Text : string; Index : Integer; DefaultColor : TColor) : TColor
11320:   Procedure CDDeleteFiles( Sender : TObject; s : string)
11321:   Function CDGetTempDir : string
11322:   Function CDGetFileSize( FileName : string) : longint
11323:   Function GetFileTime( FileName : string) : longint
11324:   Function GetShortName( FileName : string) : string
11325:   Function GetFullName( FileName : string) : string
11326:   Function WinReboot : boolean
11327:   Function WinDir : String
11328:   Function RunFile( FileToRun : string; Params : string; Dir : string; Wait : boolean) : cardinal
11329:   Function RunFile_( Cmd, WorkDir : string; Wait : boolean) : Boolean
11330:   Function devExecutor : TdevExecutor
11331: end;
11332:
11333: procedure SIRegister_FileAssocs(CL: TPSPPascalCompiler);
11334: begin
11335:   Procedure CheckAssociations // AssociationsCount', 'LongInt'( 7);
11336:   Procedure Associate( Index : integer)
11337:   Procedure UnAssociate( Index : integer)
11338:   Function IsAssociated( Index : integer) : boolean
11339:   Function CheckFiletype( const extension, filetype, description, verb, serverapp : string) : boolean
11340:   Procedure RegisterFiletype( const extension, filetype, description, verb, serverapp, IcoNum: string)
11341:   Procedure RegisterDDEServer( const filetype, verb, topic, servername, macro : string)
11342:   procedure RefreshIcons;
11343:   function GetShadeColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11344:   function MergColor(Colors: Array of TColor): TColor;
11345:   function NewColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11346:   procedure DimBitmap(ABitmap: TBitmap; Value: integer);
11347:   function GrayColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11348:   function GetInverseColor(AColor: TColor): TColor;
11349:   procedure GrayBitmap(ABitmap: TBitmap; Value: integer);
11350:   procedure DrawBitmapShadow(B: TBitmap; ACanvas: TCanvas; X,Y: integer; ShadowColor: TColor);
11351:   procedure DrawCheckMark(ACanvas: TCanvas; X, Y: integer);
11352:   Procedure GetSystemMenuFont(Font: TFont);
11353: end;
11354:
11355: //*****unit uPSI_JvHLParse;*****
11356: function IsStringConstant(const St: string): Boolean;
11357: function IsIntConstant(const St: string): Boolean;
11358: function IsRealConstant(const St: string): Boolean;
11359: function IsIdentifier(const ID: string): Boolean;
11360: function GetStringValue(const St: string): string;
11361: procedure ParseString(const S: string; Ss: TStrings);
11362: function IsStringConstantW(const St: WideString): Boolean;
11363: function IsIntConstantW(const St: WideString): Boolean;
11364: function IsRealConstantW(const St: WideString): Boolean;
11365: function IsIdentifierW(const ID: WideString): Boolean;
11366: function GetStringValueW(const St: WideString): WideString;
11367: procedure ParseStringW(const S: WideString; Ss: TStrings);
11368:
11369:
11370: //*****unit uPSI_JclMap;*****
11371:
11372: Function JclSimpleSendMail( const ARecipient, AName, ASubject, ABody : string; const AAttachment : TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean

```

```

11373: Function JclSimpleSendFax( const ARecipient, AName, ASubject, ABody : string; const AAttachment : 
11374:   TFileName; ShowDialog : Boolean; AParentWND : HWnd ) : Boolean
11375: Function JclSimpleBringUpSendMailDialog( const ASubject, ABody : string; const 
11376:   AAttach : TFileName; AParentWND : HWnd ) : Bool
11377: Function MapiCheck( const Res : DWORD; IgnoreUserAbort : Boolean ) : DWORD
11378: Function MapiErrorMessage( const ErrorCode : DWORD ) : string
11379: begin
11380:   //'Pdes_key_schedule', '^des_key_schedule // will not work
11381:   Function BuildType1Message( ADomain, AHost : String ) : String
11382:   Function BuildType3Message( ADomain, AHost, AUsername : WideString; APassword, ANonce : String ) : String
11383:   Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIAuthENTICATIONCLASS )
11384:   Function FindAuthClass( AuthName : String ) : TIAuthENTICATIONCLASS
11385: GBase64CodeTable', 'string'ABCDEFGHJKLMNOPQRSTUVWXYZabcdefghi jklmnopqrstuvwxyz0123456789+
11386: GXECODETable', 'string'+0123456789ABCDEFGHJKLMNOPQRSTUVWXYZabcdefghi jklmnopqrstuvwxyz
11387: GUUCECodeTable', 'string`^!"#$%&''()**,-./0123456789:;<=>?@ABCDEFGHIJKLMNPQRSTUVWXYZ[\]^_
11388: end;
11389:
11390: procedure SIRegister_WDOSocketUtils(CL: TPSPascalCompiler);
11391: begin
11392: ('IpAny', 'LongWord').SetUInt( $00000000 );
11393: IpLoopBack', 'LongWord').SetUInt( $7F000001 );
11394: IpBroadcast', 'LongWord').SetUInt( $FFFFFF );
11395: IpNone', 'LongWord').SetUInt( $FFFFFF );
11396: PortAny', 'LongWord( $0000 );
11397: SocketMaxConnections', 'LongInt'( 5 );
11398: TIAddr', 'LongWord
11399: TIRec', 'record IpB1 : byte; IpB2 : byte; IpB3 : byte; IpB4 : Byte; end
11400: Function HostToNetLong( HostLong : LongWord ) : LongWord
11401: Function HostToNetShort( HostShort : Word ) : Word
11402: Function NetToHostLong( NetLong : LongWord ) : LongWord
11403: Function NetToHostShort( NetShort : Word ) : Word
11404: Function StrToIp( Ip : string ) : TIAddr
11405: Function IpToStr( Ip : TIAddr ) : string
11406: end;
11407:
11408: (*-----*)
11409: procedure SIRegister_ALSMTPCClient(CL: TPSPascalCompiler);
11410: begin
11411:   TAISmtClientAuthType', '( AlSmtpClientAuthNone, alSmtpClientAuth'
11412:     +'hPlain, AlSmtpClientAuthLogin, AlSmtpClientAuthCramMD5, AlSmtpClientAuthCr'
11413:     +'amShal, AlSmtpClientAuthAutoSelect )
11414:   TAISmtClientAuthTypeSet', 'set of TAISmtClientAuthType
11415:   SIRegister_TAISmtClient(CL);
11416: end;
11417:
11418: procedure SIRegister_WDOSPlcUtils(CL: TPSPascalCompiler);
11419: begin
11420: 'TBitNo', 'Integer
11421: TStByteNo', 'Integer
11422: TStationNo', 'Integer
11423: TInOutNo', 'Integer
11424: TIO', '( EE, AA, NE, NA )
11425: TBitSet', 'set of TBitNo
11426: TAddrKind', 'set of ( akBit0, akBit1, akBit2, akOut, akNot, akBus )
11427: TBitAddrRec', 'record Kind : TAddrKind; InOutNo : TInOutNo; ByteNo : Byte; end
11428: TBitAddr', 'LongInt
11429: TByteAddrRec', 'record Kind : TAddrKind; ByteNo : Byte; end
11430: TByteAddr', 'SmallInt
11431: TInOutState', '( iosInit, iosHalt, iosRun, iosError )
11432: Function BitAddr(aIo : TIO; aInOutNo : TInOutNo; aByteNo : Byte; aBitNo : TBitNo) : TBitAddr
11433: Function BusByteAddr(aIo : TIO;aInOutNo:TInOutNo;aStat:TStatNo;aStByteNo:TStByteNo;aBitNo:TBitNo):TBitAddr;
11434: Procedure BitAddrToValues(aBitAddr:TBitAddr;var aIo:TIO;var aInOutNo:TInOutNo;var aByteNo:Byte;var
11435: aBitNo:TBitNo);
11436: Function BitAddrToStr( Value : TBitAddr ) : string
11437: Function StrToBitAddr( const Value : string ) : TBitAddr
11438: Function ByteAddr( aIo : TIO; aInOutNo : TInOutNo; aByteNo : Byte ) : TByteAddr
11439: Function BusByteAddr(aIo:TIO;aInOutNo:TInOutNo;aStation:TStationNo;aStByteNo: TStByteNo):TByteAddr
11440: Procedure ByteAddrToValues(aByteAddr:TByteAddr;var aIo:TIO;var aInOutNo:TInOutNo;var aByteNo:Byte)
11441: Function ByteAddrToStr( Value : TByteAddr ) : string
11442: Function StrToByteAddr( const Value : string ) : TByteAddr
11443: Procedure IncByteAddr( var ByteAddr : TByteAddr; Increment : Integer )
11444: Procedure DecByteAddr( var ByteAddr : TByteAddr; Decrement : Integer )
11445: Function InOutStateToStr( State : TInOutState ) : string
11446: Function MasterErrorToStr( ErrorCode : TErrorCode ) : string
11447: Function SlaveErrorToStr( ErrorCode : TErrorCode ) : string
11448:
11449: procedure SIRegister_WDOSTimers(CL: TPSPascalCompiler);
11450: begin
11451: TIntFreq', '( ifNone, if32768, if16384, if8192, if4096, if2048, '
11452:   +'if1024, if512, if256, if128, if64, if32, if16, if8, if4, if2 )
11453: DpmiPmVector', 'Int64
11454: 'DInterval', 'LongInt'( 1000 );
11455: //'DEnabled', 'Boolean')BoolToStr( True );
11456: 'DIntFreq', 'string' if64
11457: //'DMessages', 'Boolean if64';
11458: SIRegister_TwdxCustomTimer(CL);

```

```

11459:  SIRегистер_TwdxTimer(CL);
11460:  SIRегистер_TwdxRtcTimer(CL);
11461:  SIRегистер_TCustomIntTimer(CL);
11462:  SIRегистер_TIntTimer(CL);
11463:  SIRегистер_TRtcIntTimer(CL);
11464:  Function RealNow : TDateTime
11465:  Function MsToDateTime( Millisecond : LongInt ) : TDateTime
11466:  Function DateTimeToMs( Time : TDateTime ) : LongInt
11467: end;
11468:
11469: procedure SIRегистер_IdSysLogMessage(CL: TPSPascalCompiler);
11470: begin
11471:  TIIdSyslogPRI', 'Integer
11472:  TIIdSyslogFacility', '( sfKernel, sfUserLevel, sfMailSystem, sfSy'
11473:   +'stemDaemon, sfSecurityOne, sfSysLogInternal, sfLPR, sfNNTP, sfUUCP, sfCloc'
11474:   +'kDaemonOne, sfSecurityTwo, sfFTPDaemon, sfNTP, sfLogAudit, sfLogAlert, sfC'
11475:   +'lockDaemonTwo, sfLocalUseZero, sfLocalUseOne, sfLocalUseTwo, sfLocalUseThr'
11476:   +'ee, sfLocalUseFour, sfLocalUseFive, sfLocalUseSix, sfLocalUseSeven )
11477:  TIIdSyslogSeverity' '(slEmergency, slAlert, slCritical, slError, slWarning, slNotice, slInformational, slDebug)
11478:  SIRegister_TIIdSysLogMsgPart(CL);
11479:  SIRegister_TIIdSysLogMessage(CL);
11480:  Function FacilityToString( AFac : TIIdSyslogFacility ) : string
11481:  Function SeverityToString( ASec : TIIdSyslogSeverity ) : string
11482:  Function NoToSeverity( ASev : Word ) : TIIdSyslogSeverity
11483:  Function logSeverityToNo( ASev : TIIdSyslogSeverity ) : Word
11484:  Function NoToFacility( AFac : Word ) : TIIdSyslogFacility
11485:  Function logFacilityToNo( AFac : TIIdSyslogFacility ) : Word
11486: end;
11487:
11488: procedure SIRегистер_TextUtils(CL: TPSPascalCompiler);
11489: begin
11490:  'UWhitespace', 'String '(?:\s*)
11491:  Function StripSpaces( const AText : string ) : string
11492:  Function CharCount( const AText : string; Ch : Char ) : Integer
11493:  Function BalancedText( const AText : string; const Ch1, Ch2 : Char; const Count : Integer ) : string
11494:  Function BalancedTextReg( const AText:string; const Ch1, Ch2 : Char; const Count : Integer ) : string
11495: end;
11496:
11497:
11498: procedure SIRегистер_ExtPascalUtils(CL: TPSPascalCompiler);
11499: begin
11500:  ExtPascalVersion', 'String '0.9.8
11501:  AddTypeS('TBrower', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
11502:   +'Opera, brKonqueror, brMobileSafari )
11503:  AddTypeS('TCSSUnit', '( cssPX, cssPerc, cssEM, cssEX, cssIN, cssCM, cssMM, cssPT, cssPC, cssnone )
11504:  AddTypeS('TExtProcedure', 'Procedure
11505:  Function DetermineBrowser( const UserAgentStr : string ) : TBrower
11506:  Function ExtExtract(const Delims:array of string;var S:string;var Matches:TStringList;Remove:bool):bool;
11507:  Function ExtExplode( Delim : char; const S : string; Separator : char ) : TStringList
11508:  Function FirstDelimiter( const Delimiters, S : string; Offset : integer ) : integer
11509:  Function RPosEx( const Substr, Str : string; Offset : integer ) : integer
11510:  Function CountStr( const Substr, Str : string; UntilStr : string ) : integer
11511:  Function StrToJS( const S : string; UseBR : boolean ) : string
11512:  Function CaseOf( const S : string; const Cases : array of string ) : integer
11513:  Function RCaseOf( const S : string; const Cases : array of string ) : integer
11514:  Function EnumToJSString( TypeInfo : PTypeInfo; Value : integer ) : string
11515:  Function SetPaddings(Top:integer;Right:int;Bottom:intr;Left:integer;CSSUnit:TCSSUnit;Header:bool):string;
11516:  Function SetMargins(Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool): string;
11517:  Function ExtBefore( const BeforeS, AfterS, S : string ) : boolean
11518:  Function IsUpperCase( S : string ) : boolean
11519:  Function BeautifyJS(const AScript:string;const StartingLevel:integer;SplitHTMLNewLine: boolean):string;
11520:  Function BeautifyCSS( const AStyle : string ) : string
11521:  Function LengthRegExp( Rex : string; CountAll : Boolean ) : integer
11522:  Function JSDateToDate( JSDate : string ) : TDateTime
11523: end;
11524:
11525: procedure SIRегистер_JclShell(CL: TPSPascalCompiler);
11526: begin
11527:  TSHDeleteOption', '( doSilent, doAllowUndo, doFilesOnly )
11528:  TSHDeleteOptions', 'set of TSHDeleteOption
11529:  TSHRenameOption', '( roSilent, roRenameOnCollision )
11530:  TSHRenameOptions', 'set of TSHRenameOption
11531:  Function SHDeleteFiles( Parent : HWND; const Files : string; Options : TSHDeleteOptions ) : Boolean
11532:  Function SHDeleteFolder( Parent : HWND; const Folder : string; Options : TSHDeleteOptions ) : Boolean
11533:  Function SHRenameFile( const Src, Dest : string; Options : TSHRenameOptions ) : Boolean
11534:  TEnumFolderFlag', '( efFolders, efNonFolders, efiIncludeHidden )
11535:  TEnumFolderFlags', 'set of TEnumFolderFlag
11536:  TEnumFolderRec', 'record DisplayName : string; Attributes : DWOR'
11537:   +'D; IconLarge : HICON; IconSmall : HICON; Item : PItemIdList; EণumIdList : '
11538:   +'IEnumIdList; Folder : IShellFolder; end
11539:  Function SHEnumFolderFirst(const Folder:string;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Boolean;
11540:  Function SHEnumSpecialFolderFirst(SpecialFolder:DWORD;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11541:  Procedure SHenumFolderClose( var F : TEnumFolderRec)
11542:  Function SHenumFolderNext( var F : TEnumFolderRec ) : Boolean
11543:  Function GetSpecialFolderLocation( const Folder : Integer ) : string
11544:  Function DisplayPropDialog( const Handle : HWND; const FileName : string ) : Boolean;
11545:  Function DisplayPropDialog1( const Handle : HWND; const Item : PItemIdList ) : Boolean;
11546:  Function DisplayContextMenu( const Handle : HWND; const FileName : string; Pos : TPoint ) : Boolean
11547:  Function OpenFolder( const Path : string; Parent : HWND ) : Boolean

```

```

11548: Function OpenSpecialFolder( FolderID : Integer; Parent : HWND ) : Boolean
11549: Function SHReallocMem( var P : Pointer; Count : Integer ) : Boolean
11550: Function SHAllocMem( out P : Pointer; Count : Integer ) : Boolean
11551: Function SHGetMem( var P : Pointer; Count : Integer ) : Boolean
11552: Function SHFreeMem( var P : Pointer ) : Boolean
11553: Function DriveToPidlBind( const DriveName : string; out Folder : IShellFolder ) : PIItemIdList
11554: Function PathToPidl( const Path : string; Folder : IShellFolder ) : PIItemIdList
11555: Function PathToPidlBind( const FileName : string; out Folder : IShellFolder ) : PIItemIdList
11556: Function PidlBindToParent( const IdList:PIItemIdList;out Folder:IShellFolder;out Last:PIItemIdList):Bool;
11557: Function PidlCompare( const Pidl1, Pidl2 : PIItemIdList ) : Boolean
11558: Function PidlCopy( const Source : PIItemIdList; out Dest : PIItemIdList ) : Boolean
11559: Function PidlFree( var IdList : PIItemIdList ) : Boolean
11560: Function PidlGetDepth( const Pidl : PIItemIdList ) : Integer
11561: Function PidlGetLength( const Pidl : PIItemIdList ) : Integer
11562: Function PidlGetNext( const Pidl : PIItemIdList ) : PIItemIdList
11563: Function PidlToPath( Idlist : PIItemIdList ) : string
11564: Function StrRetFreeMem( StrRet : TStrRet ) : Boolean
11565: Function StrRetToString( IdList : PIItemIdList; StrRet : TStrRet; Free : Boolean ) : string
11566: PShellLink', '^TShellLink // will not work
11567: TShellLink', 'record Arguments : string; ShowCmd : Integer; Work'
11568: +'ingDirectory : string; IdList : PIItemIdList; Target : string; Description '
11569: +' : string; IconLocation : string; IconIndex : Integer; HotKey : Word; end
11570: Procedure ShellLinkFree( var Link : TShellLink )
11571: Function ShellLinkResolve( const FileName : string; var Link : TShellLink ) : HRESULT
11572: Function ShellLinkCreate( const Link : TShellLink; const FileName : string ) : HRESULT
11573: Function ShellLinkCreateSystem( const Link:TShellLink;const Folder:Integer; const FileName:string ):HRESULT;
11574: Function ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon ) : Boolean
11575: Function SHDllGetVersion( const FileName : string; var Version : TDllVersionInfo ) : Boolean
11576: Function GetSystemIcon( IconIndex : Integer; Flags : Cardinal ) : HICON
11577: Function OverlayIcon( var Icon : HICON; Overlay : HICON; Large : Boolean ) : Boolean
11578: Function OverlayIconShortCut( var Large, Small : HICON ) : Boolean
11579: Function OverlayIconShared( var Large, Small : HICON ) : Boolean
11580: Function SHGetItemInfoTip( const Folder : IShellFolder; Item : PIItemIdList ) : string
11581: Function ShellExecEx( const FileName:string;const Parameters:string;const Verb:string; CmdShow:Int ):Bool;
11582: Function ShellExec(Wnd: Integer;const Operation,FileName,Parameters,Directy:string;ShowCommand:Int ):Bool;
11583: Function ShellExecAndWait( const FileName:string;const Params:string;const Verb:string;CmdShow:Int ):Bool;
11584: Function ShellOpenAs( const FileName : string ) : Boolean
11585: Function ShellRasDial( const EntryName : string ) : Boolean
11586: Function ShellRunControlPanel( const NameOrFileName:string; AppletNumber:Integer ):Boolean
11587: Function GetFileNameIcon( const FileName : string; Flags : Cardinal ) : HICON
11588: TJclFileExeType', '( etError, etMsDos, etWin16, etWin32Gui, etWin32Con )
11589: Function GetFileExeType( const FileName : TFileName ) : TJclFileExeType
11590: Function ShellFindExecutable( const FileName, DefaultDir : string ) : string
11591: Procedure keybd_event( bvK : Byte; bScan : Byte; dwFlags, dwExtraInfo : DWORD )
11592: Function OemKeyScan( wOemChar : Word ) : DWORD
11593: Procedure mouse_event( dwFlags, dx, dy, dwData, dwExtraInfo : DWORD )
11594: end;
11595:
11596: procedure SIRegister_cXMLFunctions(CL: TPPascalCompiler);
11597: begin
11598: xmlVersion', 'String '1.0 FindClass('TOBJECT'), 'Exml
11599: //Function xmlValidChar( const Ch : AnsiChar ) : Boolean;
11600: Function xmlValidChar1( const Ch : UCS4Char ) : Boolean;
11601: Function xmlValidChar2( const Ch : WideChar ) : Boolean;
11602: Function xmlIsSpaceChar( const Ch : WideChar ) : Boolean
11603: Function xmlIsLetter( const Ch : WideChar ) : Boolean
11604: Function xmlIsDigit( const Ch : WideChar ) : Boolean
11605: Function xmlIsNameStartChar( const Ch : WideChar ) : Boolean
11606: Function xmlIsNameChar( const Ch : WideChar ) : Boolean
11607: Function xmlIsPubidChar( const Ch : WideChar ) : Boolean
11608: Function xmlValidName( const Text : UnicodeString ) : Boolean
11609: //xmlSpace', 'Char #$20 or #$9 or #$D or #$A);
11610: //Function xmlSkipSpace( var P : PWideChar ) : Boolean
11611: //Function xmlSkipEq( var P : PWideChar ) : Boolean
11612: //Function xmlExtractQuotedText( var P : PWideChar; var S : UnicodeString ) : Boolean
11613: //Function xmlGetEntityEncoding( const Buf : Pointer; const BufSize : Integer; out HeaderSize : Integer )
11614: : TUncodeCodecClass
11615: Function xmlResolveEntityReference( const RefName : UnicodeString ) : WideChar
11616: Function xmlTag( const Tag : UnicodeString ) : UnicodeString
11617: Function xmlEndTag( const Tag : UnicodeString ) : UnicodeString
11618: Function xmlAttrTag( const Tag : UnicodeString; const Attr : UnicodeString ) : UnicodeString
11619: Function xmlEmptyTag( const Tag, Attr : UnicodeString ) : UnicodeString
11620: Procedure xmlSafeTextInPlace( var Txt : UnicodeString )
11621: Function xmlSafeText( const Txt : UnicodeString ) : UnicodeString
11622: Function xmlSpaceIndent( const IndentLength : Integer; const IndentLevel : Integer ):UnicodeString
11623: Function xmlTabIndent( const IndentLevel : Integer ) : UnicodeString
11624: Function xmlComment( const Comment : UnicodeString ) : UnicodeString
11625: Procedure SelfTestcXMLFunctions
11626: end;
11627: (*-----*)
11628: procedure SIRegister_DepWalkUtils(CL: TPPascalCompiler);
11629: begin
11630: Function AWaitCursor : IUnknown
11631: Function ChangeCursor( NewCursor : TCursor ) : IUnknown
11632: Procedure SuspendRedraw( AControl : TWinControl; Suspend : boolean )
11633: Function YesNo( const ACaption, AMsg : string ) : boolean
11634: Procedure strTokenize( const S : string; Delims : TSysCharSet; Results : TStrings )
11635: Function GetBorlandLibPath( Version : integer; ForDelphi : boolean ) : string

```

```

11636: Function GetExpandedLibRoot( Version : integer; ForDelphi : boolean ) : string
11637: Procedure GetPathList( Version : integer; ForDelphi : boolean; Strings : TStrings )
11638: Procedure GetSystemPaths( Strings : TStrings )
11639: Procedure MakeEditNumeric( EditHandle : integer )
11640: end;
11641:
11642: procedure SIRegister_yuvconverts(CL: TPSPascalCompiler);
11643: begin
11644:   AddTypeS('TVideoCodec','(vcUnknown,vcRGB,vcYUY2,vcUYVY,vcBTYUV,vcYV,U9,vcYUV12,vcY8,vcY211)
11645:   'BI_YUY2','LongWord( $32595559 );
11646:   'BI_UYVY','LongWord').SetUInt( $59565955 );
11647:   'BI_BTUV','LongWord').SetUInt( $50313459 );
11648:   'BI_YVU9','LongWord').SetUInt( $39555659 );
11649:   'BI_YUV12','LongWord( $30323449 );
11650:   'BI_Y8','LongWord').SetUInt( $20203859 );
11651:   'BI_Y211','LongWord').SetUInt( $31313259 );
11652: Function BICompressionToVideoCodec( Value : DWord ) : TVideoCodec
11653: Function ConvertCodecToRGB(Codec:TVideoCodec;Src,Dst:Pointer;AWidth,AHeight:Integer):Boolean;
11654: end;
11655:
11656: (*-----*)
11657: procedure SIRegister_AviCap(CL: TPSPascalCompiler);
11658: begin
11659:   'WM_USER','LongWord').SetUInt( $0400 );
11660:   'WM_CAP_START','LongWord').SetUInt($0400);
11661:   'WM_CAP_END','longword').SetUInt($0400+85);
11662: //WM_CAP_START+ 85
11663: // WM_CAP_SET_CALLBACK_CAPCONTROL = (WM_CAP_START+ 85);
11664: Function capSetCallbackOnErrorHandler( hwnd : THandle; fpProc : LongInt ) : LongInt
11665: Function capSetCallbackOnStatus( hwnd : THandle; fpProc : LongInt ) : LongInt
11666: Function capSetCallbackOnYield( hwnd : THandle; fpProc : LongInt ) : LongInt
11667: Function capSetCallbackOnFrame( hwnd : THandle; fpProc : LongInt ) : LongInt
11668: Function capSetCallbackOnVideoStream( hwnd : THandle; fpProc : LongInt ) : LongInt
11669: Function capSetCallbackOnWaveStream( hwnd : THandle; fpProc : LongInt ) : LongInt
11670: Function capSetCallbackOnCapControl( hwnd : THandle; fpProc : LongInt ) : LongInt
11671: Function capSetUserData( hwnd : THandle; lUser : LongInt ) : Longint
11672: Function capGetUserData( hwnd : THandle ) : LongInt
11673: Function capDriverConnect( hwnd : THandle; I : Word ) : LongInt
11674: Function capDriverDisconnect( hwnd : THandle ) : LongInt
11675: Function capDriverGetName( hwnd : THandle; szName : LongInt; wSize : Word ) : LongInt
11676: Function capDriverGetVersion( hwnd : THandle; szVer : LongInt; wSize : Word ) : LongInt
11677: Function capDriverGetCaps( hwnd : THandle; s : LongInt; wSize : Word ) : LongInt
11678: Function capFileSetCaptureFile( hwnd : THandle; szName : LongInt ) : LongInt
11679: Function capFileGetCaptureFile( hwnd : THandle; szName : LongInt; wSize : Word):LongInt
11680: Function capFileAlloc( hwnd : THandle; dwSize : LongInt ) : LongInt
11681: Function capFileSaveAs( hwnd : THandle; szName : LongInt ) : LongInt
11682: Function capFileSetInfoChunk( hwnd : THandle; lpInfoChunk : LongInt ) : LongInt
11683: Function capFileSaveDIB( hwnd : THandle; szName : LongInt ) : LongInt
11684: Function capEditCopy( hwnd : THandle ) : LongInt
11685: Function capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word ) : LongInt
11686: Function capGetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word ) : LongInt
11687: Function capGetAudioFormatSize( hwnd : THandle ) : LongInt
11688: Function capDlgVideoFormat( hwnd : THandle ) : LongInt
11689: Function capDlgVideoSource( hwnd : THandle ) : LongInt
11690: Function capDlgVideoDisplay( hwnd : THandle ) : LongInt
11691: Function capDlgVideoCompression( hwnd : THandle ) : LongInt
11692: Function capGetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word ) : LongInt
11693: Function capGetVideoFormatSize( hwnd : THandle ) : LongInt
11694: Function capSetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word ) : LongInt
11695: Function capPreview( hwnd : THandle; f : Word ) : LongInt
11696: Function capPreviewRate( hwnd : THandle; wMS : Word ) : LongInt
11697: Function capOverlay( hwnd : THandle; f : Word ) : LongInt
11698: Function capOverwriteScale( hwnd : THandle; f : Word ) : LongInt
11699: Function capGetStatus( hwnd : THandle; s : LongInt; wSize : Word ) : LongInt
11700: Function capSetScrollPos( hwnd : THandle; lpP : LongInt ) : LongInt
11701: Function capGrabFrame( hwnd : THandle ) : LongInt
11702: Function capGrabFrameNoStop( hwnd : THandle ) : LongInt
11703: Function capCaptureSequence( hwnd : THandle ) : LongInt
11704: Function capCaptureSequenceNoFile( hwnd : THandle ) : LongInt
11705: Function capCaptureStop( hwnd : THandle ) : LongInt
11706: Function capCaptureAbort( hwnd : THandle ) : LongInt
11707: Function capCaptureSingleFrameOpen( hwnd : THandle ) : LongInt
11708: Function capCaptureSingleFrameClose( hwnd : THandle ) : LongInt
11709: Function capCaptureSingleFrame( hwnd : THandle ) : LongInt
11710: Function capCaptureGetSetup( hwnd : THandle; s : LongInt; wSize : Word ) : LongInt
11711: Function capCaptureSetSetup( hwnd : THandle; s : Longint; wSize : Word ) : LongInt
11712: Function capSetMCIDeviceName( hwnd : THandle; szName : LongInt ) : LongInt
11713: Function capGetMCIDeviceName( hwnd : THandle; szName : Longint; wSize : Word ) : LongInt
11714: Function capPaletteOpen( hwnd : THandle; szName : LongInt ) : LongInt
11715: Function capPaletteSave( hwnd : THandle; szName : LongInt ) : LongInt
11716: Function capPalettePaste( hwnd : THandle ) : LongInt
11717: Function capPaletteAuto( hwnd : THandle; iFrames : Word; iColors : LongInt ) : LongInt
11718: Function capPaletteManual( hwnd : THandle; fGrab : Word; iColors : LongInt ) : LongInt
11719: //PCapDriverCaps', '^TCapDriverCaps // will not work
11720: TCapDriverCaps', 'record wDeviceIndex : WORD; fHasOverlay : BOOL'
11721: +' ; fHasDlgVideoSource : BOOL; fHasDlgVideoFormat : BOOL; fHasDlgVideoDispla'
11722: +' y : BOOL; fCaptureInitialized : BOOL; fDriverSuppliesPalettes : BOOL; hVid'
11723: +'eoIn : THANDLE; hVideoOut : THANDLE; hVideoExtIn:THANDLE; hVideoExtOut:THANDLE; end
11724: //PCapStatus', '^TCapStatus // will not work

```

```

11725: TCapStatus', 'record uiImageWidth : UINT; uiImageHeight : UINT; '
11726: +'fLiveWindow : BOOL; fOverlayWindow : BOOL; fScale : BOOL; ptScroll : TPOINT'
11727: +'T; fUsingDefaultPalette : BOOL; fAudioHardware : BOOL; fCapFileExists : BO'
11728: +'OL; dwCurrentVideoFrame : DWORD; dwCurrentVideoFramesDropped : DWORD; dwCu'
11729: +'rrentWaveSamples : WORD; dwCurrentTimeElapsedMS : DWORD; hPalCurrent : HP'
11730: +'ALETTE; fCapturingNow : BOOL; dwReturn : DWORD; wNumVideoAllocated : WORD; '
11731: +' wNumAudioAllocated : WORD; end
11732: //PCaptureParms', '^TCaptureParms // will not work
11733: TCaptureParms', 'record dwRequestMicroSecPerFrame : DWORD; fMake'
11734: +'UserHitOKToCapture : BOOL; wPercentDropForError : WORD; fYield : BOOL; dwI'
11735: +'ndexSize : DWORD; wChunkGranularity : WORD; fUsingDOSMemory : BOOL; wNumVi'
11736: +'deoRequested : WORD; fCaptureAudio : BOOL; wNumAudioRequested : WORD; vKey'
11737: +'Abort : WORD; fAbortLeftMouse : BOOL; fAbortRightMouse : BOOL; fLimitEnabl'
11738: +'ed : BOOL; wTimeLimit : WORD; fMCIControl : BOOL; fStepMCIDevice : BOOL; d'
11739: +'wMCISearchBar : WORD; dwMCISearchBar : DWORD; fStepCaptureAt2x : BOOL; wSt'
11740: +'epCaptureAverageFrames : WORD; dwAudioBufferSize : DWORD; fDisableWriteCac'
11741: +'he : BOOL; AVStreamMaster : WORD; end
11742: // PCapInfoChunk', '^TCapInfoChunk // will not work
11743: //TCapInfoChunk', 'record fccInfoID : FOURCC; lpData : LongInt; cbData : LongInt; end
11744: 'CONTROLCALLBACK_PREROLL','LongInt'( 1);
11745: 'CONTROLCALLBACK_CAPTURING','LongInt'( 2);
11746: Function capCreateCaptureWindow( lpszWindowName: PChar; dwStyle : DWord; x, y : Integer; nWidth, nHeight
: Integer; hwndParent : THandle; nID : Integer ) : THandle
11747: Function
capGetDriverDescription(wDriverIndex:DWord;lpszName:PChar;cbName:Integer;lpszVer:PChar;cbVer:Int):Bool;
11748: 'IDS_CAP_BEGIN','LongInt'( 300);
11749: 'IDS_CAP_END','LongInt'( 301);
11750: 'IDS_CAP_INFO','LongInt'( 401);
11751: 'IDS_CAP_OUTOFMEM','LongInt'( 402);
11752: 'IDS_CAP_FILEEXISTS','LongInt'( 403);
11753: 'IDS_CAP_ERRORPALOPEN','LongInt'( 404);
11754: 'IDS_CAP_ERRORPALSAVE','LongInt'( 405);
11755: 'IDS_CAP_ERRORDIBSAVE','LongInt'( 406);
11756: 'IDS_CAP_DEFAVIEEXT','LongInt'( 407);
11757: 'IDS_CAP_DEFPALEXT','LongInt'( 408);
11758: 'IDS_CAP_CANTOPEN','LongInt'( 409);
11759: 'IDS_CAP_SEQ_MSGSTART','LongInt'( 410);
11760: 'IDS_CAP_SEQ_MSGSTOP','LongInt'( 411);
11761: 'IDS_CAP_VIDEODITERR','LongInt'( 412);
11762: 'IDS_CAP_READONLYFILE','LongInt'( 413);
11763: 'IDS_CAP_WRITEERROR','LongInt'( 414);
11764: 'IDS_CAP_NODISKSPACE','LongInt'( 415);
11765: 'IDS_CAP_SETFILESIZE','LongInt'( 416);
11766: 'IDS_CAP_SAVEASPERCENT','LongInt'( 417);
11767: 'IDS_CAP_DRIVER_ERROR','LongInt'( 418);
11768: 'IDS_CAP_WAVE_OPEN_ERROR','LongInt'( 419);
11769: 'IDS_CAP_WAVE_ALLOC_ERROR','LongInt'( 420);
11770: 'IDS_CAP_WAVE_PREPARE_ERROR','LongInt'( 421);
11771: 'IDS_CAP_WAVE_ADD_ERROR','LongInt'( 422);
11772: 'IDS_CAP_WAVE_SIZE_ERROR','LongInt'( 423);
11773: 'IDS_CAP_VIDEO_OPEN_ERROR','LongInt'( 424);
11774: 'IDS_CAP_VIDEO_ALLOC_ERROR','LongInt'( 425);
11775: 'IDS_CAP_VIDEO_PREPARE_ERROR','LongInt'( 426);
11776: 'IDS_CAP_VIDEO_ADD_ERROR','LongInt'( 427);
11777: 'IDS_CAP_VIDEO_SIZE_ERROR','LongInt'( 428);
11778: 'IDS_CAP_FILE_OPEN_ERROR','LongInt'( 429);
11779: 'IDS_CAP_FILE_WRITE_ERROR','LongInt'( 430);
11780: 'IDS_CAP_RECORDING_ERROR','LongInt'( 431);
11781: 'IDS_CAP_RECORDING_ERROR2','LongInt'( 432);
11782: 'IDS_CAP_AVI_INIT_ERROR','LongInt'( 433);
11783: 'IDS_CAP_NO_FRAME_CAP_ERROR','LongInt'( 434);
11784: 'IDS_CAP_NO_PALETTE_WARN','LongInt'( 435);
11785: 'IDS_CAP_MCI_CONTROL_ERROR','LongInt'( 436);
11786: 'IDS_CAP_MCI_CANT_STEP_ERROR','LongInt'( 437);
11787: 'IDS_CAP_NO_AUDIO_CAP_ERROR','LongInt'( 438);
11788: 'IDS_CAP_AVI_DRAWDIB_ERROR','LongInt'( 439);
11789: 'IDS_CAP_COMPRESSOR_ERROR','LongInt'( 440);
11790: 'IDS_CAP_AUDIO_DROP_ERROR','LongInt'( 441);
11791: 'IDS_CAP_STAT_LIVE_MODE','LongInt'( 500);
11792: 'IDS_CAP_STAT_OVERLAY_MODE','LongInt'( 501);
11793: 'IDS_CAP_STAT_CAP_INIT','LongInt'( 502);
11794: 'IDS_CAP_STAT_CAP_FINI','LongInt'( 503);
11795: 'IDS_CAP_STAT_PALETTE_BUILD','LongInt'( 504);
11796: 'IDS_CAP_STAT_OPTPAL_BUILD','LongInt'( 505);
11797: 'IDS_CAP_STAT_I_FRAMES','LongInt'( 506);
11798: 'IDS_CAP_STAT_L_FRAMES','LongInt'( 507);
11799: 'IDS_CAP_STAT_CAP_L_FRAMES','LongInt'( 508);
11800: 'IDS_CAP_STAT_CAP_AUDIO','LongInt'( 509);
11801: 'IDS_CAP_STAT_VIDEOCURRENT','LongInt'( 510);
11802: 'IDS_CAP_STAT_VIDEOAUDIO','LongInt'( 511);
11803: 'IDS_CAP_STAT_VIDEOONLY','LongInt'( 512);
11804: 'IDS_CAP_STAT_FRAMESDROPPED','LongInt'( 513);
11805: 'AVICAP32','String 'AVICAP32.dll
11806: end;
11807:
11808: procedure SIRegister_ALFcnMisc(CL: TPPSPascalCompiler);
11809: begin
11810:   Function AlBoolToInt( Value : Boolean ) : Integer
11811:   Function ALMediumPos( LTotal, LBorder, LObject : integer ) : Integer

```

```

11812: Function AlIsValidEmail( const Value : AnsiString ) : boolean
11813: Function AlLocalDateTimeToGMTDateTime( const aLocalDateTime : TDateTime ) : TdateTime
11814: Function ALInc( var x : integer; Count : integer ) : Integer
11815: function ALCopyStr(const aSourceString: AnsiString; aStart, aLength: Integer): AnsiString
11816: function ALGetStringFromFile(filename: AnsiString; const ShareMode: Word = fmShareDenyWrite):AnsiString;
11817: procedure ALSaveStringToFile(Str: AnsiString; filename: AnsiString);
11818: Function ALIsInteger(const S: AnsiString): Boolean;
11819: function ALIsDecimal(const S: AnsiString): boolean;
11820: Function ALStringToWideString(const S: AnsiString; const aCodePage: Word): WideString;
11821: function AlWideStringToString(const WS: WideString; const aCodePage: Word): AnsiString;
11822: function ALQuotedStr(const S: AnsiString; const Quote: AnsiChar = ''): AnsiString;
11823: function ALDequotedStr(const S: AnsiString; AQuote: AnsiChar): AnsiString;
11824: function ALUTF8removeBOM(const S: AnsiString): AnsiString;
11825: Function ALRandomStrL(const aLength: Longint; const aCharset: Array of Char): AnsiString;
11826: Function ALRandomStr(const aLength: Longint): AnsiString;
11827: Function ALRandomStrU1(const aLength: Longint; const aCharset: Array of Char): String;
11828: Function ALRandomStrU(const aLength: Longint): String;
11829: end;
11830:
11831: procedure SIRegister_ALJSONDoc(CL: TPPSPascalCompiler);
11832: begin
11833: Procedure ALJSONToTStrings(const AJsonStr:AnsiString;aLst:TALStrings; const aNullStr:AnsiString;const
11834: aTrueStr: AnsiString; const aFalseStr : AnsiString)
11835:
11836: procedure SIRegister_ALWindows(CL: TPPSPascalCompiler);
11837: begin
11838:   _ALMEMORYSTATUSEX', 'record dwLength : DWORD; dwMemoryLoad : DWO'
11839:     +'RD; ullTotalPhys : Int64; ullAvailPhys : Int64; ullTotalPageFile : Int64; '
11840:     +'ullAvailPageFile : Int64; ullTotalVirtual : Int64; ullAvailVirtual : Int64'
11841:     +'; ullAvailExtendedVirtual : Int64; end
11842:   TALMemoryStatusEx', '_ALMEMORYSTATUSEX
11843: Function ALGlobalMemoryStatusEx( var lpBuffer : TALMEMORYSTATUSEX ) : BOOL
11844: Function ALInterlockedExchange64( var Target : LONGLONG; Value : LONGLONG ) : LONGLONG
11845: 'INVALID_SET_FILE_POINTER','LongInt'( DWORD ( - 1 ) );
11846: 'QUOTA_LIMITS_HARDWS_MIN_DISABLE','LongWord').SetUInt( $2);
11847: 'QUOTA_LIMITS_HARDWS_MIN_ENABLE','LongWord').SetUInt( $1);
11848: 'QUOTA_LIMITS_HARDWS_MAX_DISABLE','LongWord').SetUInt( $8);
11849: 'QUOTA_LIMITS_HARDWS_MAX_ENABLE','LongWord').SetUInt( $4);
11850: end;
11851:
11852: procedure SIRegister_IPCThrd(CL: TPPSPascalCompiler);
11853: begin
11854:   SIRegister_THandledObject(CL);
11855:   SIRegister_TEvent(CL);
11856:   SIRegister_TMutex(CL);
11857:   SIRegister_TSharedMem(CL);
11858:   'TRACE_BUF_SIZE','LongInt'( 200 * 1024 );
11859:   'TRACE_BUFFER','String 'TRACE_BUFFER
11860:   'TRACE_MUTEX','String 'TRACE_MUTEX
11861:   //PTraceEntry', '^TTraceEntry // will not work
11862:   SIRegister_TIPCTracer(CL);
11863:   'MAX_CLIENTS','LongInt'( 6 );
11864:   'IPCTIMEOUT','LongInt'( 2000 );
11865:   'IPCBUFFER_NAME','String 'BUFFER_NAME
11866:   'BUFFER_MUTEX_NAME','String 'BUFFER_MUTEX
11867:   'MONITOR_EVENT_NAME','String 'MONITOR_EVENT
11868:   'CLIENT_EVENT_NAME','String 'CLIENT_EVENT
11869:   'CONNECT_EVENT_NAME','String 'CONNECT_EVENT
11870:   'CLIENT_DIR_NAME','String 'CLIENT_DIRECTORY
11871:   'CLIENT_DIR_MUTEX','String 'DIRECTORY_MUTEX
11872:   FindClass('TOBJECT'),'EMonitorActive
11873:   FindClass('TOBJECT'),'TIPCThread
11874:   TEventKind', '( evMonitorAttach, evMonitorDetach, evMonitorSigna'
11875:     +'l, evMonitorExit, evClientStart, evClientStop, evClientAttach, evClientDet'
11876:     +'ach, evClientSwitch, evClientSignal, evClientExit )
11877:   TClientFlag', '( cfError, cfMouseMove, cfMouseDown, cfResize, cfAttach )
11878:   TClientFlags', 'set of TClientFlag
11879:   //PEventData', '^TEventData // will not work
11880:   TEventData', 'record X : SmallInt; Y : SmallInt; Flag : TClientF'
11881:   +'lag; Flags : TClientFlags; end
11882:   TConnectEvent', 'Procedure ( Sender : TIPCThread; Connecting : Boolean)
11883:   TDirUpdateEvent', 'Procedure ( Sender : TIPCThread)
11884:   TIPCNotifyEvent', 'Procedure ( Sender : TIPCThread; Data : TEventData)
11885:   //PICEEventInfo', '^TIPCEEventInfo // will not work
11886:   TIPCEEventInfo', 'record FID:Integer;FKind:TEventKind;FData:TEventData;end
11887:   SIRegister_TIPCEEvent(CL);
11888:   //PCClientDirRecords', '^TClientDirRecords // will not work
11889:   SIRegister_TClientDirectory(CL);
11890:   TIPCState', '( stInactive, stDisconnected, stConnected )
11891:   SIRegister_TIPCThread(CL);
11892:   SIRegister_TIPCMonitor(CL);
11893:   SIRegister_TIPCCClient(CL);
11894:   Function IsMonitorRunning( var Hndl : THandle ) : Boolean
11895:   end;
11896:
11897: (*-----*)
11898: procedure SIRegister_ALGSMComm(CL: TPPSPascalCompiler);
11899: begin

```

```

11900:  SIRegister_TALGSMComm(CL);
11901:  Function AlGSMComm_BuildPDUMessage( aSMSCenter, aSMSAddress, aMessage : AnsiString ) : AnsiString
11902:  Procedure AlGSMComm_DecodePDUMessage(aPDUMessage:AnsiString;var aSMSCenter,aSMSAddress,
AMessage:AnsiString);
11903:  Function AlGSMComm_UnicodeToGSM7BitDefaultAlphabet( aMessage : WideString ) : AnsiString
11904:  Function AlGSMComm_GSM7BitDefaultAlphabetToUnicode(aMess:AnsiString;const
UseGreekAlphabet:Bool):Widestring;
11905:  function ALMatchesMask(const Filename, Mask: AnsiString): Boolean;
11906:  end;
11907:
11908: procedure SIRegister_ALHttpCommon(CL: TPSPascalCompiler);
11909: begin
11910:   TALHTTPPropertyChangeEvent', 'Procedure(sender:Tobject;const PropertyInfo:Integer;
11911:   TALHTTPProtocolVersion', '( HTTPPpv_1_0, HTTPPpv_1_1 )
11912:   TALHTTPMethod', '(HTTPPmt_Get,HTTPPmt_Post,HTTPPmt_Head,HTTPPmt_Trace,HTTPPmt_Put,HTTPPmt_Delete);
11913:   TIInternetScheme', 'integer
11914:   TALIPv6Binary', 'array[1..16] of Char;
11915: // TALIPv6Binary = array[1..16] of ansiChar;
11916: // TIInternetScheme = Integer;
11917:  SIRegister_TALHTTPRequestHeader(CL);
11918:  SIRegister_TALHTTPCookie(CL);
11919:  SIRegister_TALHTTPCookieCollection(CL);
11920:  SIRegister_TALHTTPResponseHeader(CL);
11921:  Function ALHTTPDecode( const AStr : AnsiString ) : AnsiString
11922:  Procedure ALHTTPEncodeParamNameValues( ParamValues : TALStrings )
11923: // Procedure ALEExtractHTTPFields(Separators,WhiteSpace, Quotes:TSysCharSet,
Content:PAnsiChar;Strings:TALStrings;StripQuotes:Boolean;
11924: // Procedure ALEExtractHeaderFields( Separators,WhiteSpace, Quotes : TSysCharSet; Content : PAnsiChar;
Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11925: // Procedure ALEExtractHeaderFieldsWithQuoteEscaped(Separators,WhiteSpace,
Quotes:TSysCharSet,Content:PAnsiChar;Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11926:  Function AlRemoveShemeFromUrl( aUrl : AnsiString ) : AnsiString
11927:  Function AlExtractShemeFromUrl( aUrl : AnsiString ) : TIInternetScheme
11928:  Function AlExtractHostNameFromUrl( aUrl : AnsiString ) : AnsiString
11929:  Function AlExtractDomainNameFromUrl( aUrl : AnsiString ) : AnsiString
11930:  Function AlExtractUrlPathFromUrl( aUrl : AnsiString ) : AnsiString
11931:  Function AlInternetCrackUrl( aUrl : AnsiString; var SchemeName,HostName,UserName,Password,UrlPath,
ExtraInfo : AnsiString; var PortNumber : integer ) : Boolean;
11932:  Function AlInternetCrackUrl1( aUrl : AnsiString; var SchemeName, HostName, UserName, Password,UrlPath,
Anchor : AnsiString; Query : TALStrings; var PortNumber : integer ) : Boolean;
11933:  Function AlInternetCrackUrl2(var Url:AnsiString;var Anchor:AnsiString;Query:TALStrings):Boolean;
11934:  Function AlRemoveAnchorFromUrl( aUrl : AnsiString; var aAnchor : AnsiString ) : AnsiString;
11935:  Function AlRemoveAnchorFromUrl1( aUrl : AnsiString ) : AnsiString;
11936:  Function AlCombineUrl( RelativeUrl, BaseUrl : AnsiString ) : AnsiString;
11937:  Function AlCombineUrl1(RelativeUrl, BaseUrl, Anchor : AnsiString; Query:TALStrings) : AnsiString;
11938:  Function ALGmtDateTimeToRfc822Str( const aValue : TDateTime ) : AnsiString
11939:  Function ALDateTimeToRfc822Str( const aValue : TDateTime ) : AnsiString
11940:  Function ALTryRfc822StrToGMTDateTime( const S : AnsiString; out Value : TDateTime ) : Boolean
11941:  Function ALRfc822StrToGMTDateTime( const s : AnsiString ) : TDateTime
11942:  Function ALTryIPV4StrToNumeric( aIPV4Str : ansiString; var aIPV4Num : Cardinal ) : Boolean
11943:  Function ALIPV4StrToNumeric( aIPV4 : ansiString ) : Cardinal
11944:  Function ALNumericToIPV4Str( aIPV4 : Cardinal ) : ansiString
11945:  Function ALZeroIpV6 : TALIPv6Binary
11946:  Function ALTryIPV6StrToBinary( aIPV6Str : ansiString; var aIPV6Bin : TALIPv6Binary ) : Boolean
11947:  Function ALIPV6StrToBinary( aIPV6 : ansiString ) : TALIPv6Binary
11948:  Function ALBinaryToIPV6Str( aIPV6 : TALIPv6Binary ) : ansiString
11949:  Function ALBinaryStrToIPV6Binary( aIPV6BinaryStr : ansiString ) : TALIPv6Binary
11950: end;
11951:
11952: procedure SIRegister_ALFcnsHTML(CL: TPSPascalCompiler);
11953: begin
11954:  Procedure ALUTF8ExtractHTMLText(HtmlCont:AnsiStr;LstExtractedResourceText:TALStrings;const
DecodeHTMLText:Bool;
11955:  Function ALUTF8ExtractHTMLText1(HtmlContent:AnsiString;const DecodeHTMLText:Boolean): AnsiString;
11956:  Function ALXMLCDataElementEncode( Src : AnsiString ) : AnsiString
11957:  Function ALXMLTextElementEncode(Src : AnsiString; const useNumericReference : boolean) : AnsiString
11958:  Function ALUTF8XMLTextElementDecode( const Src : AnsiString ) : AnsiString
11959:  Function ALUTF8HTMLEncode(const Src:AnsiStr;const EncodeASCIIHtmlEntities:Bool;const
useNumRef:bool):AnsiString;
11960:  Function ALUTF8HTMLDecode( const Src : AnsiString ) : AnsiString
11961:  Function ALJavascriptEncode( const Src : AnsiString; const useNumericReference : boolean ) : AnsiString
11962:  Function ALUTF8JavascriptDecode( const Src : AnsiString ) : AnsiString
11963:  Procedure ALHideHtmlUnwantedTagForHTMLHandleTagfunct(var HtmlContent:AnsiString; const
DeleteBodyOfUnwantedTag : Boolean; const ReplaceUnwantedTagCharBy : AnsiChar)
11964:  Procedure ALCompactHtmlTagParams( TagParams : TALStrings )
11965: end;
11966:
11967: procedure SIRegister_ALInternetMessageCommon(CL: TPSPascalCompiler);
11968: begin
11969:  SIRegister_TALEMailHeader(CL);
11970:  SIRegister_TALNewsArticleHeader(CL);
11971:  Function AlParseEmailAddress(FriendlyEmail:AnsiString;var RealName:AString;const
decodeRealName:Bool):AnsiString;
11972:  Function AlExtractEmailAddress( FriendlyEmail : AnsiString ) : AnsiString
11973:  Function ALMakeFriendlyEmailAddress( aRealName, aEmail : AnsiString ) : AnsiString
11974:  Function ALEncodeRealName4FriendlyEmailAddress( aRealName : AnsiString ) : AnsiString
11975:  Function AlGenerateInternetMessageID : AnsiString;
11976:  Function AlGenerateInternetMessageID1( ahostname : AnsiString ) : AnsiString;
11977:  Function ALDecodeQuotedPrintableString( src : AnsiString ) : AnsiString

```

```

11978: Function AlDecodeInternetMessageHeaderInUTF8(aHeaderStr:AnsiString;aDefaultCodePage:Integer):AnsiString;
11979: end;
11980:
11981: (*-----*)
11982: procedure SIRegister_ALFcnsWinSock(CL: TPSpascalCompiler);
11983: begin
11984:   Function ALHostToIP( HostName : AnsiString; var Ip : AnsiString):Boolean
11985:   Function ALIPAddrToName( IPAddr : AnsiString) : AnsiString
11986:   Function ALGetLocalIPs : TALStrings
11987:   Function ALGetLocalHostName : AnsiString
11988: end;
11989:
11990: procedure SIRegister_ALFcnsCGI(CL: TPSpascalCompiler);
11991: begin
11992:   Procedure AlCGIInitDefaultServerVariablesFromWebRequest(WebRequest :
11993:     TALWebRequest;ServerVariables:TALStrings);
11994:   Procedure AlCGIInitDefaultServerVariablesFromWebRequest1(WebRequest: TALWebRequest; ServerVariables :
11995:     TALStrings; ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
11996:   Procedure AlCGIInitDefaultServerVariables( ServerVariables : TALStrings);
11997:   Procedure AlCGIInitDefaultServerVariables1(ServerVars:TALStrings;ScriptName,
11998:     ScriptFileName:AnsiString;Url:AnsiStr;
11999:   Procedure AlCGIInitServerVariablesFromWebRequest( WebRequest:TALWebRequest; ServerVariables : TALStrings;
12000:     ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
12001:   Procedure AlCGIExec( InterpreterFilename : AnsiString; ServerVariables : TALStrings; RequestContentStream
12002:     : Tstream; ResponseContentStream : Tstream; ResponseHeader : TALHTTPResponseHeader);
12003:   Procedure AlCGIExec1( ScriptName,ScriptFileName, Url, X_REWRITE_URL, InterpreterFilename:AnsiString;
12004:     WebRequest : TALIsapiRequest;
12005:     overloadedCookies:AnsiString;overloadedQueryString:AnsiString;overloadedReferer: AnsiString;
12006:     +'overloadedRequestContentStream:Tstream;var
12007:     ResponseContentStr:AnsiString;ResponseHeader:TALHTTPResponseHeader;
12008:   Procedure AlCGIExec2( ScriptName,ScriptFileName,Url,X_REWRITE_URL,
12009:     InterpreterFilename:AnsiString;WebRequest: TALIsapiRequest; var ResponseContentString : AnsiString;
12010:     ResponseHeader : TALHTTPResponseHeader);
12011: end;
12012:
12013: procedure SIRegister_ALFcnsExecute(CL: TPSpascalCompiler);
12014: begin
12015:   TStartupInfoA', 'TStartupInfo
12016:   'SE_CREATE_TOKEN_NAME','String'( 'SeCreateTokenPrivilege
12017:   SE_ASSIGNPRIMARYTOKEN_NAME','String' 'SeAssignPrimaryTokenPrivilege
12018:   SE_LOCK_MEMORY_NAME','String'( 'SeLockMemoryPrivilege
12019:   SE_INCREASE_QUOTA_NAME','String' 'SeIncreaseQuotaPrivilege
12020:   SE_UNSOLICITED_INPUT_NAME','String' 'SeUnsolicitedInputPrivilege
12021:   SE_MACHINE_ACCOUNT_NAME','String' 'SeMachineAccountPrivilege
12022:   SE_TCB_NAME','String' 'SeTcbPrivilege
12023:   SE_SECURITY_NAME','String' 'SeSecurityPrivilege
12024:   SE_TAKE_OWNERSHIP_NAME','String' 'SeTakeOwnershipPrivilege
12025:   SE_LOAD_DRIVER_NAME','String' 'SeLoadDriverPrivilege
12026:   SE_SYSTEM_PROFILE_NAME','String' 'SeSystemProfilePrivilege
12027:   SE_SYSTEMTIME_NAME','String' 'SeSystemtimePrivilege
12028:   SE_PROF_SINGLE_PROCESS_NAME','String' 'SeProfileSingleProcessPrivilege
12029:   SE_INC_BASE_PRIORITY_NAME','String' 'SeIncreaseBasePriorityPrivilege
12030:   SE_CREATE_PAGEFILE_NAME','String' 'SeCreatePagefilePrivilege
12031:   SE_CREATE_PERMANENT_NAME','String' 'SeCreatePermanentPrivilege
12032:   SE_BACKUP_NAME','String' 'SeBackupPrivilege
12033:   SE_RESTORE_NAME','String' 'SeRestorePrivilege
12034:   SE_SHUTDOWN_NAME','String' 'SeShutdownPrivilege
12035:   SE_DEBUG_NAME','String' 'SeDebugPrivilege
12036:   SE_AUDIT_NAME','String' 'SeAuditPrivilege
12037:   SE_SYSTEM_ENVIRONMENT_NAME','String' 'SeSystemEnvironmentPrivilege
12038:   SE_CHANGE_NOTIFY_NAME','String' 'SeChangeNotifyPrivilege
12039:   SE_REMOTE_SHUTDOWN_NAME','String' 'SeRemoteShutdownPrivilege
12040:   SE_UNDOCK_NAME','String' 'SeUndockPrivilege
12041:   SE_SYNC_AGENT_NAME','String' 'SeSyncAgentPrivilege
12042:   SE_ENABLE_DELEGATION_NAME','String' 'SeEnableDelegationPrivilege
12043:   SE_MANAGE_VOLUME_NAME','String' 'SeManageVolumePrivilege
12044:   Function AlGetEnvironmentString : AnsiString
12045:   Function ALWinExec32(const FileName,CurrentDir,
12046:     Environment:AnsiString;InStream:Tstream;OutStream:TStream):Dword;
12047:   Function ALWinExec321(const FileName:AnsiString; InputStream:Tstream;OutputStream:TStream):Dword;
12048:   Function ALWinExecAndWait32( FileName : AnsiString; Visibility : integer ) : DWORD
12049:   Function ALWinExecAndWait32V2( FileName : AnsiString; Visibility : integer ) : DWORD
12050:   Function ALNTSetPrivilege( sPrivilege : AnsiString; bEnabled : Boolean ) : Boolean
12051: end;
12052:
12053: procedure SIRegister_ALFcnsFile(CL: TPSpascalCompiler);
12054: begin
12055:   Function AlEmptyDirectory(Directory:ansiString;SubDirectory:Bool;IgnoreFiles:array of AnsiString; const
12056:     RemoveEmptySubDirectory : Boolean; const FileNameMask : ansiString; const MinFileAge : TdateTime):Boolean;
12057:   Function AlEmptyDirectory1( Directory : ansiString; SubDirectory : Boolean; const
12058:     RemoveEmptySubdirectory:Bool; const FileNameMask : ansiString; const MinFileAge : TdateTime ) : Boolean;
12059:   Function AlCopyDirectory( SrcDirectory, DestDirectory : ansiString; SubDirectory : Boolean; const
12060:     FileNameMask : ansiString; const FailIfExists : Boolean ) : Boolean
12061:   Function ALGetModuleName : ansistring
12062:   Function ALGetModuleFileNameWithoutExtension : ansistring
12063:   Function ALGetModulePath : ansistring
12064:   Function AlGetFileSize( const AFileName : ansistring ) : int64
12065:   Function AlGetFileVersion( const AFileName : ansistring ) : ansistring
12066:   Function ALGetFileCreationDateTime( const aFileName : Ansistring ) : TDate
12067:
12068:
12069:
12070:
12071:
12072:
12073:
12074:
12075:
12076:
12077:
12078:
12079:
12080:
12081:
12082:
12083:
12084:
12085:
12086:
12087:
12088:
12089:
12090:
12091:
12092:
12093:
12094:
12095:
12096:
12097:
12098:
12099:
12100:
12101:
12102:
12103:
12104:
12105:
12106:
12107:
12108:
12109:
12110:
12111:
12112:
12113:
12114:
12115:
12116:
12117:
12118:
12119:
12120:
12121:
12122:
12123:
12124:
12125:
12126:
12127:
12128:
12129:
12130:
12131:
12132:
12133:
12134:
12135:
12136:
12137:
12138:
12139:
12140:
12141:
12142:
12143:
12144:
12145:
12146:
12147:
12148:
12149:
12150:
12151:
12152:
12153:
12154:
12155:
12156:
12157:
12158:
12159:
12160:
12161:
12162:
12163:
12164:
12165:
12166:
12167:
12168:
12169:
12170:
12171:
12172:
12173:
12174:
12175:
12176:
12177:
12178:
12179:
12180:
12181:
12182:
12183:
12184:
12185:
12186:
12187:
12188:
12189:
12190:
12191:
12192:
12193:
12194:
12195:
12196:
12197:
12198:
12199:
12200:
12201:
12202:
12203:
12204:
12205:
12206:
12207:
12208:
12209:
12210:
12211:
12212:
12213:
12214:
12215:
12216:
12217:
12218:
12219:
12220:
12221:
12222:
12223:
12224:
12225:
12226:
12227:
12228:
12229:
12230:
12231:
12232:
12233:
12234:
12235:
12236:
12237:
12238:
12239:
12240:
12241:
12242:
12243:
12244:
12245:
12246:
12247:
12248:
12249:
12250:
12251:
12252:
12253:
12254:
12255:
12256:
12257:
12258:
12259:
12260:
12261:
12262:
12263:
12264:
12265:
12266:
12267:
12268:
12269:
12270:
12271:
12272:
12273:
12274:
12275:
12276:
12277:
12278:
12279:
12280:
12281:
12282:
12283:
12284:
12285:
12286:
12287:
12288:
12289:
12290:
12291:
12292:
12293:
12294:
12295:
12296:
12297:
12298:
12299:
12300:
12301:
12302:
12303:
12304:
12305:
12306:
12307:
12308:
12309:
12310:
12311:
12312:
12313:
12314:
12315:
12316:
12317:
12318:
12319:
12320:
12321:
12322:
12323:
12324:
12325:
12326:
12327:
12328:
12329:
12330:
12331:
12332:
12333:
12334:
12335:
12336:
12337:
12338:
12339:
12340:
12341:
12342:
12343:
12344:
12345:
12346:
12347:
12348:
12349:
12350:
12351:
12352:
12353:
12354:
12355:
12356:
12357:
12358:
12359:
12360:
12361:
12362:
12363:
12364:
12365:
12366:
12367:
12368:
12369:
12370:
12371:
12372:
12373:
12374:
12375:
12376:
12377:
12378:
12379:
12380:
12381:
12382:
12383:
12384:
12385:
12386:
12387:
12388:
12389:
12390:
12391:
12392:
12393:
12394:
12395:
12396:
12397:
12398:
12399:
12400:
12401:
12402:
12403:
12404:
12405:
12406:
12407:
12408:
12409:
12410:
12411:
12412:
12413:
12414:
12415:
12416:
12417:
12418:
12419:
12420:
12421:
12422:
12423:
12424:
12425:
12426:
12427:
12428:
12429:
12430:
12431:
12432:
12433:
12434:
12435:
12436:
12437:
12438:
12439:
12440:
12441:
12442:
12443:
12444:
12445:
12446:
12447:
12448:
12449:
12450:
12451:
12452:
12453:
12454:
12455:
12456:
12457:
12458:
12459:
12460:
12461:
12462:
12463:
12464:
12465:
12466:
12467:
12468:
12469:
12470:
12471:
12472:
12473:
12474:
12475:
12476:
12477:
12478:
12479:
12480:
12481:
12482:
12483:
12484:
12485:
12486:
12487:
12488:
12489:
12490:
12491:
12492:
12493:
12494:
12495:
12496:
12497:
12498:
12499:
12500:
12501:
12502:
12503:
12504:
12505:
12506:
12507:
12508:
12509:
12510:
12511:
12512:
12513:
12514:
12515:
12516:
12517:
12518:
12519:
12520:
12521:
12522:
12523:
12524:
12525:
12526:
12527:
12528:
12529:
12530:
12531:
12532:
12533:
12534:
12535:
12536:
12537:
12538:
12539:
12540:
12541:
12542:
12543:
12544:
12545:
12546:
12547:
12548:
12549:
12550:
12551:
12552:
12553:
12554:
12555:
12556:
12557:
12558:
12559:
12560:
12561:
12562:
12563:
12564:
12565:
12566:
12567:
12568:
12569:
12570:
12571:
12572:
12573:
12574:
12575:
12576:
12577:
12578:
12579:
12580:
12581:
12582:
12583:
12584:
12585:
12586:
12587:
12588:
12589:
12590:
12591:
12592:
12593:
12594:
12595:
12596:
12597:
12598:
12599:
12600:
12601:
12602:
12603:
12604:
12605:
12606:
12607:
12608:
12609:
12610:
12611:
12612:
12613:
12614:
12615:
12616:
12617:
12618:
12619:
12620:
12621:
12622:
12623:
12624:
12625:
12626:
12627:
12628:
12629:
12630:
12631:
12632:
12633:
12634:
12635:
12636:
12637:
12638:
12639:
12640:
12641:
12642:
12643:
12644:
12645:
12646:
12647:
12648:
12649:
12650:
12651:
12652:
12653:
12654:
12655:
12656:
12657:
12658:
12659:
12660:
12661:
12662:
12663:
12664:
12665:
12666:
12667:
12668:
12669:
12670:
12671:
12672:
12673:
12674:
12675:
12676:
12677:
12678:
12679:
12680:
12681:
12682:
12683:
12684:
12685:
12686:
12687:
12688:
12689:
12690:
12691:
12692:
12693:
12694:
12695:
12696:
12697:
12698:
12699:
12700:
12701:
12702:
12703:
12704:
12705:
12706:
12707:
12708:
12709:
12710:
12711:
12712:
12713:
12714:
12715:
12716:
12717:
12718:
12719:
12720:
12721:
12722:
12723:
12724:
12725:
12726:
12727:
12728:
12729:
12730:
12731:
12732:
12733:
12734:
12735:
12736:
12737:
12738:
12739:
12740:
12741:
12742:
12743:
12744:
12745:
12746:
12747:
12748:
12749:
12750:
12751:
12752:
12753:
12754:
12755:
12756:
12757:
12758:
12759:
12760:
12761:
12762:
12763:
12764:
12765:
12766:
12767:
12768:
12769:
12770:
12771:
12772:
12773:
12774:
12775:
12776:
12777:
12778:
12779:
12780:
12781:
12782:
12783:
12784:
12785:
12786:
12787:
12788:
12789:
12790:
12791:
12792:
12793:
12794:
12795:
12796:
12797:
12798:
12799:
12800:
12801:
12802:
12803:
12804:
12805:
12806:
12807:
12808:
12809:
12810:
12811:
12812:
12813:
12814:
12815:
12816:
12817:
12818:
12819:
12820:
12821:
12822:
12823:
12824:
12825:
12826:
12827:
12828:
12829:
12830:
12831:
12832:
12833:
12834:
12835:
12836:
12837:
12838:
12839:
12840:
12841:
12842:
12843:
12844:
12845:
12846:
12847:
12848:
12849:
12850:
12851:
12852:
12853:
12854:
12855:
12856:
12857:
12858:
12859:
12860:
12861:
12862:
12863:
12864:
12865:
12866:
12867:
12868:
12869:
12870:
12871:
12872:
12873:
12874:
12875:
12876:
12877:
12878:
12879:
12880:
12881:
12882:
12883:
12884:
12885:
12886:
12887:
12888:
12889:
12890:
12891:
12892:
12893:
12894:
12895:
12896:
12897:
12898:
12899:
12900:
12901:
12902:
12903:
12904:
12905:
12906:
12907:
12908:
12909:
12910:
12911:
12912:
12913:
12914:
12915:
12916:
12917:
12918:
12919:
12920:
12921:
12922:
12923:
12924:
12925:
12926:
12927:
12928:
12929:
12930:
12931:
12932:
12933:
12934:
12935:
12936:
12937:
12938:
12939:
12940:
12941:
12942:
12943:
12944:
12945:
12946:
12947:
12948:
12949:
12950:
12951:
12952:
12953:
12954:
12955:
12956:
12957:
12958:
12959:
12960:
12961:
12962:
12963:
12964:
12965:
12966:
12967:
12968:
12969:
12970:
12971:
12972:
12973:
12974:
12975:
12976:
12977:
12978:
12979:
12980:
12981:
12982:
12983:
12984:
12985:
12986:
12987:
12988:
12989:
12990:
12991:
12992:
12993:
12994:
12995:
12996:
12997:
12998:
12999:
12999:
13000:
13001:
13002:
13003:
13004:
13005:
13006:
13007:
13008:
13009:
130010:
130011:
130012:
130013:
130014:
130015:
130016:
130017:
130018:
130019:
130020:
130021:
130022:
130023:
130024:
130025:
130026:
130027:
130028:
130029:
130030:
130031:
130032:
130033:
130034:
130035:
130036:
130037:
130038:
130039:
130040:
130041:
130042:
130043:
130044:
130045:
130046:
130047:
130048:
130049:
130050:
130051:
130052:
130053:
130054:
130055:
130056:
130057:
130058:
130059:
130060:
130061:
130062:
130063:
130064:
130065:
130066:
130067:
130068:
130069:
130070:
130071:
130072:
130073:
130074:
130075:
130076:
130077:
130078:
130079:
130080:
130081:
130082:
130083:
130084:
130085:
130086:
130087:
130088:
130089:
130090:
130091:
130092:
130093:
130094:
130095:
130096:
130097:
130098:
130099:
1300100:
1300101:
1300102:
1300103:
1300104:
1300105:
1300106:
1300107:
1300108:
1300109:
1300110:
1300111:
1300112:
1300113:
1300114:
1300115:
1300116:
1300117:
1300118:
1300119:
1300120:
1300121:
1300122:
1300123:
1300124:
1300125:
1300126:
1300127:
1300128:
1300129:
1300130:
1300131:
1300132:
1300133:
1300134:
1300135:
1300136:
1300137:
1300138:
1300139:
1300140:
1300141:
1300142:
1300143:
1300144:
1300145:
1300146:
1300147:
1300148:
1300149:
1300150:
1300151:
1300152:
1300153:
1300154:
1300155:
1300156:
1300157:
1300158:
1300159:
1300160:
1300161:
1300162:
1300163:
1300164:
1300165:
1300166:
1300167:
1300168:
1300169:
1300170:
1300171:
1300172:
1300173:
1300174:
1300175:
1300176:
1300177:
1300178:
1300179:
1300180:
1300181:
1300182:
1300183:
1300184:
1300185:
1300186:
1300187:
1300188:
1300189:
1300190:
1300191:
1300192:
1300193:
1300194:
1300195:
1300196:
1300197:
1300198:
1300199:
1300200:
1300201:
1300202:
1300203:
1300204:
1300205:
1300206:
1300207:
1300208:
1300209:
1300210:
1300211:
1300212:
1300213:
1300214:
1300215:
1300216:
1300217:
1300218:
1300219:
1300220:
1300221:
1300222:
1300223:
1300224:
1300225:
1300226:
1300227:
1300228:
1300229:
1300230:
1300231:
1300232:
1300233:
1300234:
1300235:
1300236:
1300237:
1300238:
1300239:
1300240:
1300241:
1300242:
1300243:
1300244:
1300245:
1300246:
1300247:
1300248:
1300249:
1300250:
1300251:
1300252:
1300253:
1300254:
1300255:
1300256:
1300257:
1300258:
1300259:
1300260:
1300261:
1300262:
1300263:
1300264:
1300265:
1300266:
1300267:
1300268:
1300269:
1300270:
1300271:
1300272:
1300273:
1300274:
1300275:
1300276:
1300277:
1300278:
1300279:
1300280:
1300281:
1300282:
1300283:
1300284:
1300285:
1300286:
1300287:
1300288:
1300289:
1300290:
1300291:
1300292:
1300293:
1300294:
1300295:
1300296:
1300297:
1300298:
1300299:
1300299:
1300300:
1300301:
1300302:
1300303:
1300304:
1300305:
1300306:
1300307:
1300308:
1300309:
1300309:
1300310:
1300311:
1300312:
1300313:
1300314:
1300315:
1300316:
1300317:
1300318:
1300319:
1300319:
1300320:
1300321:
1300322:
1300323:
1300324:
1300325:
1300326:
1300327:
1300328:
1300329:
1300329:
1300330:
1300331:
1300332:
1300333:
1300334:
1300335:
1300336:
1300337:
1300338:
1300339:
1300339:
1300340:
1300341:
1300342:
1300343:
1300344:
1300345:
1300346:
1300347:
1300348:
1300349:
1300349:
1300350:
1300351:
1300352:
1300353:
1300354:
1300355:
1300356:
1300357:
1300358:
1300359:
1300359:
1300360:
1300361:
1300362:
1300363:
1300364:
1300365:
1300366:
1300367:
1300368:
1300369:
1300369:
1300370:
1300371:
1300372:
1300373:
1300374:
1300375:
1300376:
1300377:
1300378:
1300379:
1300379:
1300380:
1300381:
1300382:
1300383:
1300384:
1300385:
1300386:
1300387:
1300388:
1300389:
1300389:
1300390:
1300391:
1300392:
1300393:
1300394:
1300395:
1300396:
1300397:
1300398:
1300399:
1300399:
1300400:
1300401:
1300402:
1300403:
1300404:
1300405:
1300406:
1300407:
1300408:
1300409:
1300409:
1300410:
1300411:
1300412:
1300413:
1300414:
1300415:
1300416:
1300417:
1300418:
1300419:
1300419:
1300420:
1300421:
1300422:
1300423:
1300424:
1300425:
1300426:
1300427:
1300428:
1300429:
1300429:
1300430:
1300431:
1300432:
1300433:
1300434:
1300435:
1300436:
1300437:
1300438:
1300439:
1300439:
1300440:
1300441:
1300442:
1300443:
1300444:
1300445:
1300446:
13
```

```

12053: Function ALGetFileLastWriteDateTime( const aFileName : Ansistring ) : TDateTime
12054: Function ALGetFileLastAccessDateTime( const aFileName : Ansistring ) : TDateTime
12055: Procedure ALSetFileCreationDateTime( const aFileName : Ansistring; const aCreationDateTime : TDateTime )
12056: Function ALIsDirectoryEmpty( const directory : ansiString ) : boolean
12057: Function ALFileExists( const Path : ansistring ) : boolean
12058: Function ALDirectoryExists( const Directory : Ansistring ) : Boolean
12059: Function ALCreateDir( const Dir : Ansistring ) : Boolean
12060: Function ALRemoveDir( const Dir : Ansistring ) : Boolean
12061: Function ALDeleteFile( const FileName : Ansistring ) : Boolean
12062: Function ALRenameFile( const OldName, NewName : ansistring ) : Boolean
12063: end;
12064:
12065: procedure SIRegister_ALFcnMime(CL: TPPSPascalCompiler);
12066: begin
12067:   NativeInt', 'Integer
12068:   NativeUInt', 'Cardinal
12069:   Function ALMimeBase64EncodeString( const S : AnsiString ) : AnsiString
12070:   Function ALMimeBase64EncodeStringNoCRLF( const S : AnsiString ) : AnsiString
12071:   Function ALMimeBase64DecodeString( const S : AnsiString ) : AnsiString
12072:   Function ALMimeBase64EncodedSize( const InputSize : NativeInt ) : NativeInt
12073:   Function ALMimeBase64EncodedSizeNoCRLF( const InputSize : NativeInt ) : NativeInt
12074:   Function ALMimeBase64DecodedSize( const InputSize : NativeInt ) : NativeInt
12075:   Procedure ALMimeBase64Encode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt )
12076:   Procedure ALMimeBase64EncodeNoCRLF( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt )
12077:   Procedure ALMimeBase64EncodeFullLines( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt )
12078:   Function ALMimeBase64Decode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt ) : NativeInt;
12079:   Function ALMimeBase64DecodePartial( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt );
12080:   + 'var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12081:   Function ALMimeBase64DecodePartialEnd( out OutputBuffer : TByteDynArray; OutputOffset : NativeInt; const
ByteBuffer : Cardinal; const ByteBufferSpace : Cardinal) : NativeInt;
12082:   Procedure ALMimeBase64Encode( const InputBuf:TByteDynArray;const InputByteCnt:NatInt;out
OutputBuf:TByteDynArray );
12083:   Procedure ALMimeBase64EncodeNoCRLF( const InputBuffer:TByteDynArray; const InputByteCount:NativeInt;out
OutputBuffer:TByteDynArray );
12084:   Procedure ALMimeBase64EncodeFullLines( const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
OutputBuffer:TByteDynArray );
12085:   Function ALMimeBase64Decode( const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
OutputBuffer:TByteDynArray ):NativeInt;
12086:   Function ALMimeBase64DecodePartial( const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
OutputBuffer: TByteDynArray; var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal ) : NativeInt;
12087:   Function ALMimeBase64DecodePartialEnd( out OutputBuffer:TByteDynArray;const ByteBuffer:Cardinal;const
ByteBufferSpace:Cardinal ):NativeInt;
12088:   Procedure ALMimeBase64EncodeFile( const InputFileName, OutputFileName : TFileName )
12089:   Procedure ALMimeBase64EncodeFileNoCRLF( const InputFileName, OutputFileName : TFileName )
12090:   Procedure ALMimeBase64DecodeFile( const InputFileName, OutputFileName : TFileName )
12091:   Procedure ALMimeBase64EncodeStream( const InputStream : TStream; const OutputStream : TStream )
12092:   Procedure ALMimeBase64EncodeStreamNoCRLF( const InputStream : TStream; const OutputStream : TStream )
12093:   Procedure ALMimeBase64DecodeStream( const InputStream : TStream; const OutputStream : TStream )
12094:   +'cALMimeBase64_ENCODED_LINE_BREAK','LongInt'( 76 );
12095:   +'cALMimeBase64_DECODED_LINE_BREAK','LongInt'( cALMimeBase64_ENCODED_LINE_BREAK div 4 * 3 );
12096:   +'cALMimeBase64_BUFFER_SIZE','LongInt'( cALMimeBase64_DECODED_LINE_BREAK * 3 * 4 * 4 );
12097:   Procedure ALFillMimeContentTypeByExtList( AMIMELIST : TALStrings )
12098:   Procedure ALFillExtByMimeTypeList( AMIMELIST : TALStrings )
12099:   Function ALGetDefaultFileExtFromMimeType( aContentType : AnsiString ) : AnsiString
12100:   Function ALGetDefaultMIMEContentTypeFromExt( aExt : AnsiString ) : AnsiString
12101: end;
12102:
12103: procedure SIRegister_ALXmlDoc(CL: TPPSPascalCompiler);
12104: begin
12105:   'cALXMLNodeMaxListSize','LongInt'( Maxint div 16 );
12106:   FindClass( 'TOBJECT' ), 'TALXMLNode
12107:   FindClass( 'TOBJECT' ), 'TALXMLNodeList
12108:   FindClass( 'TOBJECT' ), 'TALXMLDocument
12109:   TALXMLParseProcessingInstructionEvent', 'Procedure ( Sender:TObject; const Target,Data:Ansistring )
12110:   TALXMLParseTextEvent', 'Procedure ( Sender : TObject; const str: Ansistring )
12111:   TALXMLParseStartElementEvent', 'Procedure ( Sender : TObject; co'
12112:   +'nst Name : AnsiString; const Attributes : TALStrings )
12113:   TALXMLParseEndElementEvent', 'Procedure ( Sender : TObject; const Name : AnsiString )
12114:   TALXMLNodeType', '( ntReserved, ntElement, ntAttribute, ntText, '
12115:   +'ntCDATA, ntEntityRef, ntEntity, ntProcessingInstr, ntComment, ntDocument, '
12116:   +'ntDocType, ntDocFragment, ntNotation )
12117:   TALXMLDocOption', '( doNodeAutoCreate, doNodeAutoIndent )
12118:   TALXMLDocOptions', 'set of TALXMLDocOption
12119:   TALXMLParseOption', '( poPreserveWhiteSpace, poIgnoreXMLReferences )
12120:   TALXMLParseOptions', 'set of TALXMLParseOption
12121:   TALXMLPrologItem', '( xpVersion, xpEncoding, xpStandalone )
12122:   PALPointerXMLNodeList', '^PALPointerXMLNodeList // will not work
12123:   SIRegister_PALXMLDocError(CL);
12124:   SIRegister_TALXMLNodeList(CL);
12125:   SIRegister_TALXMLNode(CL);
12126:   SIRegister_TALXmlElementNode(CL);
12127:   SIRegister_TALXmlAttributeNode(CL);
12128:   SIRegister_TALXmlTextNode(CL);
12129:   SIRegister_TALXmlDocumentNode(CL);

```

```

12130: SIRегистер_TALXmlCommentNode(CL);
12131: SIRегистер_TALXmlProcessingInstrNode(CL);
12132: SIRегистер_TALXmlCDataNode(CL);
12133: SIRегистер_TALXmlEntityRefNode(CL);
12134: SIRегистер_TALXmlEntityNode(CL);
12135: SIRегистер_TALXmlDocTypeNode(CL);
12136: SIRегистер_TALXmlDocFragmentNode(CL);
12137: SIRегистер_TALXmlNotationNode(CL);
12138: SIRегистер_TALXMLDocument(CL);
12139: cALXMLUTF8EncodingStr', 'String 'UTF-8
12140: cALXMLUTF8HeaderStr', 'String' <?xmlversion="1.0"encoding="UTF-8"standalone="yes"?>' + #13#10);
12141: CALNSDelim', 'String ';
12142: CALXML', 'String 'xml
12143: CALVersion', 'String 'version
12144: CALEncoding', 'String 'encoding
12145: CALStandalone', 'String 'standalone
12146: CALDefaultNodeIndent', 'String '
12147: CALXmlDocument', 'String 'DOCUMENT
12148: Function ALCreateEmptyXMLDocument( const Rootname : AnsiString ) : TalXMLDocument
12149: Procedure ALClearXMLDocument( const rootname:AnsiString;xmldoc:TalXMLDocument;const
EncodingStr:AnsiString );
12150: Function ALFindXmlNodeByChildNodeValue(xmlrec:TalxmlNode;const ChildnodeName,
ChildnodeValue:AnsiString;const Recurse: Boolean) : TalxmlNode
12151: Function ALFindXmlNodeByNameAndChildNodeValue( xmlrec : TalxmlNode; const NodeName : ansiString; const
ChildnodeName, ChildnodeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12152: Function ALFindXmlNodeByAttribute(xmlrec:TalxmlNode;const AttributeName,AttributeValue:AnsiString;const
Recurse: Boolean):TalxmlNode
12153: Function ALFindXmlNodeByNameAndAttribute( xmlrec : TalxmlNode; const NodeName : ansiString; const
AttributeName,AttributeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12154: Function ALExtractAttrValue( const AttrName, AttrLine : AnsiString; const Default : AnsiString ) :
AnsiString
12155: end;
12156:
12157: procedure SIRегистер_TeCanvas(CL: TPSPascalCompiler);
12158: //based on TEEProc, TeCanvas, TEEEngine, TChart
12159: begin
12160: 'TeePiStep','Double').setExtended( Pi / 180.0 );
12161: 'TeeDefaultPerspective','LongInt'( 100 );
12162: 'TeeMinAngle','LongInt'( 270 );
12163: 'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ) );
12164: 'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ) );
12165: 'teeclCream','LongWord( TColor ( $FOFBFF ) );
12166: 'teeclMedGray','LongWord').SetUInt( TColor ( $A4AOAO ) );
12167: 'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ) );
12168: 'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ) );
12169: 'teeclCream','LongWord').SetUInt( TColor ( $FOFBFF ) );
12170: 'teeclMedGray','LongWord').SetUInt( TColor ( $A4AOAO ) );
12171: 'TA_LEFT','LongInt'( 0 );
12172: 'TA_RIGHT','LongInt'( 2 );
12173: 'TA_CENTER','LongInt'( 6 );
12174: 'TA_TOP','LongInt'( 0 );
12175: 'TA_BOTTOM','LongInt'( 8 );
12176: 'teePATCOPY','LongInt'( 0 );
12177: 'NumCirclePoints','LongInt'( 64 );
12178: 'teeDEFAULT_CHARSET','LongInt'( 1 );
12179: 'teeANTIALIASED_QUALITY','LongInt'( 4 );
12180: 'TA_LEFT','LongInt'( 0 );
12181: 'bs_Solid','LongInt'( 0 );
12182: 'teepf24Bit','LongInt'( 0 );
12183: 'teepfDevice','LongInt'( 1 );
12184: 'CM_MOUSELEAVE','LongInt'( 10000 );
12185: 'CM_SYSCOLORCHANGE','LongInt'( 10001 );
12186: 'DC_BRUSH','LongInt'( 18 );
12187: 'DC_PEN','LongInt'( 19 );
12188: teeCOLORREF', 'LongWord
12189: TLogBrush', 'record lbStyle : Integer; lbColor : TColor; lbHatch: Integer; end
12190: //TNotifyEvent', 'Procedure ( Sender : TObject )
12191: SIRегистер_TFilterRegion(CL);
12192: SIRегистер_IFormCreator(CL);
12193: SIRегистер_TTeeFilter(CL);
12194: //TFilterClass', 'class of TTeeFilter
12195: SIRегистер_TFilterItems(CL);
12196: SIRегистер_TConvolveFilter(CL);
12197: SIRегистер_TBblurFilter(CL);
12198: SIRегистер_TTeePicture(CL);
12199: TPenEndStyle', '( esRound, esSquare, esFlat )
12200: SIRегистер_TChartPen(CL);
12201: SIRегистер_TChartHiddenPen(CL);
12202: SIRегистер_TDottedGrayPen(CL);
12203: SIRегистер_TDarkGrayPen(CL);
12204: SIRегистер_TWhitePen(CL);
12205: SIRегистер_TChartBrush(CL);
12206: TTeeView3DSrolled', 'Procedure ( IsHoriz : Boolean)
12207: TTeeView3DChangedZoom', 'Procedure ( NewZoom : Integer)
12208: SIRегистер_TVview3DOptions(CL);
12209: FindClass('TOBJECT'),TTeeCanvas
12210: TTeeTransparency', 'Integer
12211: SIRегистер_TTeeBlend(CL);
12212: FindClass('TOBJECT'),TCanvas3D

```

```

12213: SIRegister_TTeeShadow(CL);
12214: teeTGradientDirection', '(gdTopBottom, gdBottomTop, gdLeftRight, gdRightLeft, gdFromCenter, gdFromTopLeft,
12215: gdFromBottomLeft, gdRadial, gdDiagonalUp, gdDiagonalDown')
12216: FindClass('TOBJECT'), 'TSubGradient
12217: SIRegister_TCustomTeeGradient(CL);
12218: SIRegister_TSubGradient(CL);
12219: SIRegister_TTeeGradient(CL);
12220: SIRegister_TTeeFontGradient(CL);
12221: TCanvasBackMode', '(cbmNone, cbmTransparent, cbmOpaque)
12222: TCanvasTextAlign', 'Integer
12223: TTeeCanvasHandle', 'HDC
12224: SIRegister_TTeeCanvas(CL);
12225: TPoint3DFloat', 'record X : Double; Y : Double; Z : Double; end
12226: SIRegister_TFloatXYZ(CL);
12227: TPoint3D', 'record x : integer; y : integer; z : Integer; end
12228: TRGB', 'record blue: byte; green: byte; red: byte; end
12229: {TRGB=packed record
12230:   Blue : Byte;
12231:   Green : Byte;
12232:   Red : Byte;
12233: //$/IFDEF CLX //Alpha : Byte; // Linux end;}
12234:
12235: TTeeCanvasCalcPoints', 'Function ( x, z : Integer; var P0, P1 :
12236:   +'TPoint3D; var Color0, Color1 : TColor) : Boolean
12237: TTeeCanvasSurfaceStyle', '(tcsSolid, tcsWire, tcsDot )
12238: TCanvas3DPlane', '( cpX, cpY, cpZ )
12239: //IInterface', 'IUnknown
12240: SIRegister_TCanvas3D(CL);
12241: SIRegister_TTeeCanvas3D(CL);
12242: TTrianglePoints', 'Array[0..2] of TPoint;
12243: TFourPoints', 'Array[0..3] of TPoint;
12244: Function ApplyDark( Color : TColor; HowMuch : Byte) : TColor
12245: Function ApplyBright( Color : TColor; HowMuch : Byte) : TColor
12246: Function Point3D( const x, y, z : Integer) : TPoint3D
12247: Procedure SwapDouble( var a, b : Double)
12248: Procedure SwapInteger( var a, b : Integer)
12249: Procedure RectSize( const R : TRect; var RectWidth, RectHeight : Integer)
12250: Procedure teeRectCenter( const R : TRect; var X, Y : Integer)
12251: Function RectFromPolygon( const Points : array of TPoint; NumPoints : Integer): TRect
12252: Function RectFromTriangle( const Points : TTrianglePoints) : TRect
12253: Function RectangleInRectangle( const Small, Big : TRect) : Boolean
12254: Procedure ClipCanvas( ACanvas : TCanvas; const Rect : TRect)
12255: Procedure UnClipCanvas( ACanvas : TCanvas)
12256: Procedure ClipEllipse( ACanvas : TTeeCanvas; const Rect : TRect)
12257: Procedure ClipRoundRectangle(ACanvas:TTeeCanvas;const Rect : TRect; RoundSize : Integer)
12258: Procedure ClipPolygon(ACanvas:TTeeCanvas;const Points:array of TPoint;NumPoints:Integer)
12259: 'TeeCharForHeight', 'String 'W
12260: 'DarkerColorQuantity', 'Byte').SetUInt( 128);
12261: 'DarkColorQuantity', 'Byte').SetUInt( 64);
12262: TButtonGetColorProc', 'Function : TColor
12263: SIRegister_TTeeButton(CL);
12264: SIRegister_TButtonColor(CL);
12265: SIRegister_TComboFlat(CL);
12266: Procedure TeeSetTeePen(FPen:TPen; APen : TChartPen; AColor : TColor; Handle:TTeeCanvasHandle)
12267: Function TeePoint( const ax, ay : Integer) : TPoint
12268: Function TEEPointInRect( const Rect : TRect; const P : TPoint) : Boolean;
12269: Function PointInRectl( const Rect : TRect; x, y : Integer) : Boolean;
12270: Function TeeRect( Left, Top, Right, Bottom : Integer) : TRect
12271: Function OrientRectangle( const R : TRect) : TRect
12272: Procedure TeeSetBitmapSize( Bitmap : TBitmap; Width, Height : Integer)
12273: Function PolygonBounds( const P : array of TPoint) : TRect
12274: Function PolygonInPolygon( const A, B : array of TPoint) : Boolean
12275: Function RGBValue( const Color : TColor) : TRGB
12276: Function EditColor( AOwner : TComponent; AColor : TColor) : TColor
12277: Function EditColorDialog( AOwner : TComponent; var AColor : TColor) : Boolean
12278: Function PointAtDistance( AFrom, ATo : TPoint; ADist : Integer) : TPoint
12279: Function TeeCull( const P : TFourPoints) : Boolean;
12280: Function TeeCull1( const P0, P1, P2 : TPoint) : Boolean;
12281: TSmoothStretchOption', '( ssBestQuality, ssBestPerformance )
12282: Procedure SmoothStretch( Src, Dst : TBitmap);
12283: Procedure SmoothStretch1( Src, Dst : TBitmap; Option : TSmoothStretchOption);
12284: Function TeeDistance( const x, y : Double) : Double
12285: Function TeeLoadLibrary( const FileName : String) : HInst
12286: Procedure TeeFreeLibrary( hLibModule : HMODULE)
12287: Procedure TeeBlendBitmaps( const Percent : Double; ABitmap, BBitmap : TBitmap; BOrigin : TPoint)
12288: //Procedure TeeCalcLines( var Lines : TRGBArray; Bitmap : TBitmap)
12289: Procedure TeeShadowSmooth(Bitmap, Back : TBitmap; Left, Top, Width, Height, horz, vert : Integer;
12290: Smoothness : Double; FullDraw : Boolean; ACanvas : TCanvas3D; Clip : Boolean)
12291: SIRegister_ICanvasHyperlinks(CL);
12292: Function Supports( const Instance : IInterface; const IID : TGUID) : Boolean
12293: end;
12294:
12295: procedure SIRegister_ovcmisc(CL: TPPSPascalCompiler);
12296: begin
12297:   TOvcHdc', 'Integer
12298:   TOvcHWND', 'Cardinal
12299:   TOvcHdc', 'HDC

```

```

12300:  TOvcHWNDF', 'HWND
12301:  Function LoadBaseBitmap( lpBitmapName : PChar ) : HBITMAP
12302:  Function LoadBaseCursor( lpCursorName : PChar ) : HCURSOR
12303:  Function ovComStruct( const S1, S2, Size : Cardinal ) : Integer
12304:  Function DefaultEpoch : Integer
12305:  Function DrawButtonFrame(Canvas:TCanvas;const Client:TRect;IsDown,IsFlat:Bool;Style:TButtonStyle):TRect;
12306:  Procedure FixRealPrim( P : PChar; DC : Char )
12307:  Function GetDisplayString( Canvas : TCanvas; const S : string; MinChars, MaxWidth : Integer ) : string
12308:  Function GetLeftButton : Byte
12309:  Function GetNextDlgItem( Ctrl : TOvcHWnd ) : hWnd
12310:  Procedure GetRGB( Clr : TColor; var IR, IG, IB : Byte )
12311:  Function GetShiftFlags : Byte
12312:  Function ovCreateRotatedFont( F : TFont; Angle : Integer ) : hFont
12313:  Function GetTopTextMargin(Font:TFont;BorderStyle:TBorderStyle;Height:Integer;Ctl3D:Boolean): Integer
12314:  Function ovExtractWord( N : Integer; const S : string; WordDelims : TCharSet ) : string
12315:  Function ovIsForegroundTask : Boolean
12316:  Function ovTrimLeft( const S : string ) : string
12317:  Function ovTrimRight( const S : string ) : string
12318:  Function ovQuotedStr( const S : string ) : string
12319:  Function ovWordCount( const S : string; const WordDelims : TCharSet ) : Integer
12320:  Function ovWordPosition(const N:Integer;const S: string;const WordDelims : TCharSet) : Integer
12321:  Function PtrDiff( const P1, P2 : PChar ) : Word
12322:  Procedure PtrInc( var P, Delta : Word )
12323:  Procedure PtrDec( var P, Delta : Word )
12324:  Procedure FixTextBuffer( InBuf, OutBuf : PChar; OutSize : Integer )
12325:  Procedure TransStretchBlt( DstDC : TOvcHdc; DstX, DstY, DstW, DstH : Integer; SrcDC : TOvcHdc; SrcX, SrcY,
      SrcW, SrcH : Integer; MaskDC : TOvcHdc; MaskX, MaskY : Integer )
12326:  Function ovMinI( X, Y : Integer ) : Integer
12327:  Function ovMaxI( X, Y : Integer ) : Integer
12328:  Function ovMinL( X, Y : LongInt ) : LongInt
12329:  Function ovMaxL( X, Y : LongInt ) : LongInt
12330:  Function GenerateComponentName( PF : TWinControl; const Root : string ) : string
12331:  Function PartialCompare( const S1, S2 : string ) : Boolean
12332:  Function PathEllipsis( const S : string; MaxWidth : Integer ) : string
12333:  Function ovCreateDisabledBitmap( FOriginal : TBitmap; OutlineColor : TColor ) : TBitmap
12334:  Procedure ovCopyParentImage( Control : TControl; Dest : TCanvas )
12335:  Procedure ovDrawTransparentBitmap( Dest : TCanvas; X, Y, W, H : Integer; Rect : TRect; Bitmap : TBitmap;
      TransparentColor : TColor )
12336:  Procedure DrawTransparentBitmapPrim(DC:TOvcHdc;Bitmap:HBitmap;xStart,yStart,Width:Int;Rect:TRect;
      TransparentColor : TColorRef)
12337:  Function ovWidthOf( const R : TRect ) : Integer
12338:  Function ovHeightOf( const R : TRect ) : Integer
12339:  Procedure ovDebugOutput( const S : string )
12340:  Function GetArrowWidth( Width, Height : Integer ) : Integer
12341:  Procedure StripCharSeq( CharSeq : string; var Str : string )
12342:  Procedure StripCharFromEnd( aChr : Char; var Str : string )
12343:  Procedure StripCharFromFront( aChr : Char; var Str : string )
12344:  Function SystemParametersInfo( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT ) : BOOL
12345:  Function SystemParametersInfoNCM(uiAction,uiParam:UINT;pvParam:TNonClientMetrics;fWinIni:UINT):BOOL;
12346:  Function SystemParametersInfoA( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT ) : BOOL
12347:  Function CreateEllipticRgn( p1, p2, p3, p4 : Integer ) : HRGN
12348:  Function CreateEllipticRgnIndirect( const p1 : TRect ) : HRGN
12349:  Function CreateFontIndirect( const p1 : TLogFont ) : HFONT
12350:  Function CreateMetaFile( p1 : PChar ) : HDC
12351:  Function DescribePixelFormat(DC: HDC;p2:Int;p3:UINT;var p4:TPixelFormatDescriptor): BOOL
12352:  Function DrawText(hdc: HDC;lpString:PChar;nCount:Integer; var lpRect : TRect;uFormat:UINT):Integer
12353:  Function DrawTextS(hdc: HDC;lpString:string;nCount:Integer; var lpRect: TRect;uFormat:UINT):Integer
12354:  Function SetMapperFlags( DC : HDC; Flag : DWORD ) : DWORD
12355:  Function SetGraphicsMode( hdc : HDC; iMode : Integer ) : Integer
12356:  Function SetMapMode( DC : HDC; p2 : Integer ) : Integer
12357:  Function SetMetaFileBitsEx( Size : UINT; const Data : PChar ) : HMETAFILE
12358: //Function SetPaletteEntries(Palette:HPalette;startIndex,NumEntries:UINT,var PaletteEntries):UINT
12359:  Function SetPixel( DC : HDC; X, Y : Integer; Color : COLORREF ) : COLORREF
12360:  Function SetPixelV( DC : HDC; X, Y : Integer; Color : COLORREF ) : BOOL
12361: //Function SetPixelFormat( DC : HDC; PixelFormat : Integer; FormatDef : PPixelFormatDescriptor ) : BOOL
12362:  Function SetPolyFillMode( DC : HDC; PolyFillMode : Integer ) : Integer
12363:  Function StretchBlt(DestDC:HDC;X,Y,Width,Height:Int;SrcDC:HDC;XSrc,YSrc,SrcWidth,
      SrcHeight:Int;Rop:DWORD):BOOL
12364:  Function SetRectRgn( Rgn : HRgn; X1, Y1, X2, Y2 : Integer ) : BOOL
12365:  Function StretchDIBits( DC : HDC; DestX,DestY,DestWidth,DestHeight,SrcX,SrcY,SrcWidth,
      SrcHeight:Int;Bits:int; var BitsInfo : TBitmapInfo; Usage : UINT; Rop : DWORD ) : Integer
12366:  Function SetROP2( DC : HDC; p2 : Integer ) : Integer
12367:  Function SetStretchBltMode( DC : HDC; StretchMode : Integer ) : Integer
12368:  Function SetSystemPaletteUse( DC : HDC; p2 : UINT ) : UINT
12369:  Function SetTextCharacterExtra( DC : HDC; CharExtra : Integer ) : Integer
12370:  Function SetTextColor( DC : HDC; Color : COLORREF ) : COLORREF
12371:  Function SetTextAlign( DC : HDC; Flags : UINT ) : UINT
12372:  Function SetTextJustification( DC : HDC; BreakExtra, BreakCount : Integer ) : Integer
12373:  Function UpdateColors( DC : HDC ) : BOOL
12374:  Function GetViewportExtEx( DC : HDC; var Size : TSize ) : BOOL
12375:  Function GetViewportOrgEx( DC : HDC; var Point : TPoint ) : BOOL
12376:  Function GetWindowExtEx( DC : HDC; var Size : TSize ) : BOOL
12377:  Function GetWindowOrgEx( DC : HDC; var Point : TPoint ) : BOOL
12378:  Function IntersectClipRect( DC : HDC; X1, Y1, X2, Y2 : Integer ) : Integer
12379:  Function InvertRgn( DC : HDC; p2 : HRGN ) : BOOL
12380:  Function MaskBlt(DestDC:HDC; XDest,YDest,Width,Height:Int;SrcDC : HDC; XScr, YScr : Integer; Mask :
      HBITMAP; xMask, yMask : Integer; Rop : DWORD ) : BOOL
12381:  Function PlgBlt(DestDC:HDC;const PtsArray,SrcDC:HDC;XSrc,YSrc,Widt,Heigh:Int;Mask:HBITMAP;xMask,
      yMask:Int):BOOL;

```

```

12382: Function OffsetClipRgn( DC : HDC; XOffset, YOffset : Integer ) : Integer
12383: Function OffsetRgn( RGN : HRGN; XOffset, YOffset : Integer ) : Integer
12384: Function PatBlt( DC : HDC; X, Y, Width, Height : Integer; Rop : DWORD ) : BOOL
12385: Function Pie( DC : HDC; X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Integer ) : BOOL
12386: Function PlayMetaFile( DC : HDC; MF : HMETAFILE ) : BOOL
12387: Function PaintRgn( DC : HDC; RGN : HRGN ) : BOOL
12388: Function PtInRegion( RGN : HRGN; X, Y : Integer ) : BOOL
12389: Function PtVisible( DC : HDC; X, Y : Integer ) : BOOL
12390: Function RectInRegion( RGN : HRGN; const Rect : TRect ) : BOOL
12391: Function RectVisible( DC : HDC; const Rect : TRect ) : BOOL
12392: Function Rectangle( DC : HDC; X1, Y1, X2, Y2 : Integer ) : BOOL
12393: Function RestoreDC( DC : HDC; SavedDC : Integer ) : BOOL
12394: end;
12395:
12396: procedure SIRegister_ovcfiler(CL: TPSPPascalCompiler);
12397: begin
12398:   SIRegister_TOvcAbstractStore(CL);
12399:   // PExPropInfo', '^TExPropInfo // will not work
12400: //  TExPropInfo', 'record PI : TPropInfo; AObject : TObject; end
12401:   SIRegister_TOvcPropertyList(CL);
12402:   SIRegister_TOvcDataFiler(CL);
12403:   Procedure UpdateStoredList( AForm : TWinControl; AStoredList : TStrings; FromForm : Boolean )
12404:   Procedure UpdateStoredList1( AForm : TCustomForm; AStoredList : TStrings; FromForm : Boolean )
12405:   Function CreateStoredItem( const CompName, PropName : string ) : string
12406:   Function ParseStoredItem( const Item : string; var CompName, PropName : string ) : Boolean
12407: //Function GetPropType( PropInfo : PExPropInfo ) : PTypeInfo
12408: end;
12409:
12410: procedure SIRegister_ovccoco(CL: TPSPPascalCompiler);
12411: begin
12412:   'ovsetsize','LongInt'( 16 );
12413:   'etSyntax','LongInt'( 0 );
12414:   'etSemantic','LongInt'( 1 );
12415:   'chCR','Char #13';
12416:   'chLF','Char #10';
12417:   'chLineSeparator',' chCR );
12418:   SIRegister_TCocoError(CL);
12419:   SIRegister_TCommentItem(CL);
12420:   SIRegister_TCommentList(CL);
12421:   SIRegister_TSymbolPosition(CL);
12422:   TGenListType', '( glNever, glAlways, glOnError )
12423:   TovBitSet', 'set of Integer
12424:   //PStartTable', '^TStartTable // will not work
12425:   'TovCharSet', 'set of AnsiChar
12426:   TAAfterGenListEvent', 'Procedure ( Sender : TObject; var PrintErrorCount : boolean)
12427:   TCommentEvent', 'Procedure ( Sender : TObject; CommentList : TCommentList )
12428:   TCustomErrorEvent', 'Function(Sender:TObject;const ErrorCode: longint;const Data:string): string
12429:   TovErrorEvent', 'Procedure ( Sender : TObject; Error : TCocoError )
12430:   TovErrorProc', 'Procedure ( ErrorCode : integer; Symbol : TSymbolP'
12431:     +'osition; const Data : string; ErrorType : integer)
12432:   TFailureEvent', 'Procedure ( Sender : TObject; NumErrors : integer )
12433:   TGetCH', 'Function( pos : longint ) : char
12434:   TStatusUpdateProc', 'Procedure ( Sender : TObject; Status: string; LineNum:integer )
12435:   SIRegister_TCocoRScanner(CL);
12436:   SIRegister_TCocoRGrammar(CL);
12437:   '_EF','Char #0);
12438:   '_TAB','Char').SetString( #09);
12439:   '_CR','Char').SetString( #13);
12440:   '_LF','Char').SetString( #10);
12441:   '_EL','').SetString( _CR);
12442:   '_EOF','Char').SetString( #26);
12443:   'LineEnds','TCharSet'(ord(_CR) or ord(_LF) or ord(_EF));
12444:   'minErrDist','LongInt'( 2 );
12445:   Function ovPadL( S : string; ch : char; L : integer ) : string
12446: end;
12447:
12448: TFormatSettings = record
12449:   CurrencyFormat: Byte;
12450:   NegCurrFormat: Byte;
12451:   ThousandSeparator: Char;
12452:   DecimalSeparator: Char;
12453:   CurrencyDecimals: Byte;
12454:   DateSeparator: Char;
12455:   TimeSeparator: Char;
12456:   ListSeparator: Char;
12457:   CurrencyString: string;
12458:   ShortDateFormat: string;
12459:   LongDateFormat: string;
12460:   TimeAMString: string;
12461:   TimePMString: string;
12462:   ShortTimeFormat: string;
12463:   LongTimeFormat: string;
12464:   ShortMonthNames: array[1..12] of string;
12465:   LongMonthNames: array[1..12] of string;
12466:   ShortDayNames: array[1..7] of string;
12467:   LongDayNames: array[1..7] of string;
12468:   TwoDigitYearCenturyWindow: Word;
12469: end;
12470:

```

```

12471: procedure SIRegister_OvcFormatSettings(CL: TPPSPascalCompiler);
12472: begin
12473:   Function ovFormatSettings : TFormatSettings
12474: end;
12475:
12476: procedure SIRegister_ovcstr(CL: TPPSPascalCompiler);
12477: begin
12478:   TOvcCharSet', 'set of Char
12479:   ovBTable', 'array[0..255] of Byte
12480:   //BTable = array[0..{$IFDEF UNICODE}{$HUGE_UNICODE_BMTABLE}{$FFFF}{$ELSE}{$FF}{$ENDIF}{$ENDIF}]{${$IFDEF UNICODE}{$HUGE_UNICODE_BMTABLE}{$FFFF}{$ELSE}{$FF}{$ENDIF}{$ENDIF}} of Byte;
12481:   Function BinaryBPChar( Dest : PChar; B : Byte ) : PChar
12482:   Function BinaryLPChar( Dest : PChar; L : LongInt ) : PChar
12483:   Function BinaryWPChar( Dest : PChar; W : Word ) : PChar
12484:   Procedure BMMakeTable( MatchString : PChar; var BT : ovBTable )
12485:   Function BMSearch(var Buffer,BufLength:Cardinal;var BT:ovBTable;MatchString:PChar;var Pos:Cardinal):Bool;
12486:   Function BMSearchUC(var Buffer,BufLength:Card; var BT:ovBTable;MatchString:PChar; var Pos: Card):Boolean;
12487:   Function CharStrPChar( Dest : PChar; C : Char; Len : Cardinal ) : PChar
12488:   Function DetabPChar( Dest : PChar; Src : PChar; TabSize : Byte ) : PChar
12489:   Function HexBPChar( Dest : PChar; B : Byte ) : PChar
12490:   Function HexLPChar( Dest : PChar; L : LongInt ) : PChar
12491:   Function HexPtrPChar( Dest : PChar; P : TObject ) : PChar
12492:   Function HexWPChar( Dest : PChar; W : Word ) : PChar
12493:   Function LoCaseChar( C : Char ) : Char
12494:   Function OctalLPChar( Dest : PChar; L : LongInt ) : PChar
12495:   Function StrChDeletePrim( P : PChar; Pos : Cardinal ) : PChar
12496:   Function StrChInsertPrim( Dest : PChar; C : Char; Pos : Cardinal ) : PChar
12497:   Function StrChPos( P : PChar; C : Char; var Pos : Cardinal ) : Boolean
12498:   Procedure StrInsertChars( Dest : PChar; Ch : Char; Pos, Count : Word )
12499:   Function StrStCopy( Dest, S : PChar; Pos, Count : Cardinal ) : PChar
12500:   Function StrStDeletePrim( P : PChar; Pos, Count : Cardinal ) : PChar
12501:   Function StrStInsert( Dest, S1, S2 : PChar; Pos : Cardinal ) : PChar
12502:   Function StrStInsertPrim( Dest, S : PChar; Pos : Cardinal ) : PChar
12503:   Function StrStPos( P, S : PChar; var Pos : Cardinal ) : Boolean
12504:   Function StrToInt64PChar( S : PChar; var I : LongInt ) : Boolean
12505:   Procedure TrimAllSpacesPChar( P : PChar )
12506:   Function TrimEmbeddedZeros( const S : string ) : string
12507:   Procedure TrimEmbeddedZerosPChar( P : PChar )
12508:   Function TrimTrailPrimPChar( S : PChar ) : PChar
12509:   Function TrimTrailPChar( Dest, S : PChar ) : PChar
12510:   Function TrimTrailingZeros( const S : string ) : string
12511:   Procedure TrimTrailingZerosPChar( P : PChar )
12512:   Function UpCaseChar( C : Char ) : Char
12513:   Function ovcCharInSet( C : Char; const CharSet : TOvcCharSet ) : Boolean
12514:   Function ovc32StringIsCurrentCodePage( const S : WideString ) : Boolean;
12515:   //Function ovc32StringIsCurrentCodePage1( const S:PWideChar; CP : Cardinal ) : Boolean;
12516: end;
12517:
12518: procedure SIRegister_AfUtils(CL: TPPSPascalCompiler);
12519: begin
12520:   //PRaiseFrame', '^TRaiseFrame // will not work
12521:   TRaiseFrame', 'record NextRaise : PRaiseFrame; ExceptAddr : __Poin'
12522:   +'ter; ExceptObject : TObject; ExceptionRecord : PExceptionRecord; end
12523:   Procedure SafeCloseHandle( var Handle : THandle )
12524:   Procedure ExchangeInteger( X1, X2 : Integer )
12525:   Procedure FillInteger( const Buffer, Size, Value : Integer )
12526:   Function LongMulDiv( Multi, Mult2, Divl : Longint ) : Longint
12527:   Function afCompareMem( P1, P2 : TObject; Length : Integer ) : Boolean
12528:
12529: FILENAME_ADVAPI32      = 'ADVAPI32.DLL';
12530:   function AbortSystemShutdown; external advapi32 name 'AbortSystemShutdownW';
12531: function AbortSystemShutdown(lpMachineName: PKOLOChar): BOOL; stdcall;
12532:   function AccessCheckAndAuditAlarm(SubsystemName: PKOLOChar;
12533:   HandleId: Pointer; ObjectTypeName, ObjectName: PKOLOChar;
12534:   SecurityDescriptor: PSecurityDescriptor; DesiredAccess: DWORD;
12535:   const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12536:   var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12537:   function AccessCheckByTypeAndAuditAlarm(SubsystemName: PKOLOChar;
12538:   HandleId: Pointer; ObjectTypeName, ObjectName: PKOLOChar;
12539:   SecurityDescriptor: PSecurityDescriptor; PrincipalsSelfSid: PSID; DesiredAccess: DWORD;
12540:   AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12541:   ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12542:   var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12543:   function AccessCheckByTypeResultListAndAuditAlarm(SubsystemName: PKOLOChar;
12544:   HandleId: Pointer; ObjectTypeName, ObjectName: PKOLOChar;
12545:   SecurityDescriptor: PSecurityDescriptor; PrincipalsSelfSid: PSID; DesiredAccess: DWORD;
12546:   AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12547:   ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12548:   var GrantedAccess: DWORD; var AccessStatusList:DWORD; var pfGenerateOnClose: BOOL): BOOL; stdcall;
12549:   function BackupEventLog(hEventLog: THandle; lpBackupFileName: PKOLOChar): BOOL; stdcall;
12550:   function ClearEventLog(hEventLog: THandle; lpBackupFileName: PKOLOChar): BOOL; stdcall;
12551:   function CreateProcessAsUser(hToken: THandle; lpApplicationName: PKOLOChar;
12552:   lpCommandLine: PKOLOChar; lpProcessAttributes: PSecurityAttributes;
12553:   lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL;
12554:   dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PKOLOChar;
12555:   const lpStartupInfo: TStartupInfo; var lpProcessInformation: TProcessInformation): BOOL; stdcall;
12556:   function GetCurrentHwProfile(var lphwProfileInfo: THWProfileInfo): BOOL; stdcall;
12557:   function GetFileSecurity(lpFileName: PKOLOChar; RequestedInformation: SECURITY_INFORMATION;
12558:   pSecurityDescriptor: PSecurityDescriptor;nLength:DWORD;var lpnLengthNeeded: DWORD):BOOL; stdcall;

```

```

12559:     function GetUserName(lpBuffer: PKOOLChar; var nSize: DWORD): BOOL; stdcall;
12560:     function InitiateSystemShutdown(lpMachineName, lpMessage: PKOOLChar;
12561:         dwTimeOut: DWORD; bForceAppsClosed, bRebootAfterShutdown: BOOL): BOOL; stdcall;
12562:     function LogonUser(lpszUsername, lpszDomain, lpszPassword: PKOOLChar;
12563:         dwLogonProvider: DWORD; var phToken: THandle): BOOL; stdcall;
12564:     function LookupAccountName(lpSystemName, lpAccountName: PKOOLChar;
12565:         Sid: PSID; var cbSid: DWORD; ReferencedDomainName: PKOOLChar;
12566:             var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12567:     function LookupAccountSid(lpSystemName: PKOOLChar; Sid: PSID;
12568:         Name: PKOOLChar; var cbName: DWORD; ReferencedDomainName: PKOOLChar;
12569:             var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12570:     function LookupPrivilegeDisplayName(lpSystemName, lpName: PKOOLChar;
12571:         lpDisplayName: PKOOLChar; var cbDisplayName, lpLanguageId: DWORD): BOOL; stdcall;
12572:     function LookupPrivilegeName(lpSystemName: PKOOLChar;
12573:         var lpLuid: TLargeInteger; lpName: PKOOLChar; var cbName: DWORD): BOOL; stdcall;
12574:     function LookupPrivilegeValue(lpSystemName, lpName: PKOOLChar;
12575:         var lpLuid: TLargeInteger): BOOL; stdcall;
12576:     function ObjectCloseAuditAlarm(SubsystemName: PKOOLChar;
12577:         HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12578:     function ObjectDeleteAuditAlarm(SubsystemName: PKOOLChar;
12579:         HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12580:     function ObjectOpenAuditAlarm(SubsystemName: PKOOLChar; HandleId: Pointer;
12581:         ObjectTypeName: PKOOLChar; ObjectName: PKOOLChar; pSecurityDescriptor: PSecurityDescriptor;
12582:         ClientToken: THandle; DesiredAccess, GrantedAccess: DWORD;
12583:             var Privileges: TPrivilegeSet; ObjectCreation, AccessGranted: BOOL;
12584:             var GenerateOnClose: BOOL): BOOL; stdcall;
12585:     function ObjectPrivilegeAuditAlarm(SubsystemName: PKOOLChar;
12586:         HandleId: Pointer; ClientToken: THandle; DesiredAccess: DWORD;
12587:             var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12588:     function OpenBackupEventLog(lpUNCServerName, lpFileName: PKOOLChar): THandle; stdcall;
12589:     function OpenEventLog(lpUNCServerName, lpSourceName: PKOOLChar): THandle; stdcall;
12590:     function PrivilegedServiceAuditAlarm(SubsystemName, ServiceName: PKOOLChar;
12591:         ClientToken: THandle; var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12592:     function ReadEventLog(hEventLog: THandle; dwReadFlags, dwRecordOffset: DWORD;
12593:         lpBuffer: Pointer; nNumberOfBytesToRead: DWORD;
12594:             var pnBytesRead, pnMinNumberofBytesNeeded: DWORD): BOOL; stdcall;
12595:     function RegConnectRegistry(lpMachineName: PKOOLChar; hKey: HKEY;
12596:         var phkResult: HKEY): Longint; stdcall;
12597:     function RegCreateKey(hKey: HKEY; lpSubKey: PKOOLChar;
12598:         var phkResult: HKEY): Longint; stdcall;
12599:     function RegCreateKeyEx(hKey: HKEY; lpSubKey: PKOOLChar;
12600:         Reserved: DWORD; lpClass: PKOOLChar; dwOptions: DWORD; samDesired: REGSAM;
12601:             lpSecurityAttributes: PSecurityAttributes; var phkResult: HKEY;
12602:                 lpdwDisposition: PDWORD): Longint; stdcall;
12603:     function RegDeleteKey(hKey: HKEY; lpSubKey: PKOOLChar): Longint; stdcall;
12604:     function RegDeleteValue(hKey: HKEY; lpValueName: PKOOLChar): Longint; stdcall;
12605:     function RegEnumKeyEx(hKey: HKEY; dwIndex: DWORD; lpName: PKOOLChar;
12606:         var lpcbName: DWORD; lpReserved: Pointer; lpClass: PKOOLChar;
12607:             lpcbClass: PDWORD; lpftLastWriteTime: PFileTime): Longint; stdcall;
12608:     function RegEnumKey(hKey: HKEY; dwIndex: DWORD; lpName: PKOOLChar; cbName: DWORD): Longint; stdcall;
12609:     function RegEnumValue(hKey: HKEY; dwIndex: DWORD; lpValueName: PKOOLChar;
12610:         var lpcbValueName: DWORD; lpReserved: Pointer; lpType: PDWORD;
12611:             lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12612:     function RegLoadKey(hKey: HKEY; lpSubKey, lpFile: PKOOLChar): Longint; stdcall;
12613:     function RegOpenKey(hKey: HKEY; lpSubKey: PKOOLChar; var phkResult: HKEY): Longint; stdcall;
12614:     function RegOpenKeyEx(hKey: HKEY; lpSubKey: PKOOLChar;
12615:         ulOptions: DWORD; samDesired: REGSAM; var phkResult: HKEY): Longint; stdcall;
12616:     function RegQueryInfoKey(hKey: HKEY; lpClass: PKOOLChar;
12617:         lpcbClass: PDWORD; lpReserved: Pointer;
12618:             lpcbSubKeys, lpcbMaxSubKeyLen, lpcbMaxClassLen, lpcValues,
12619:                 lpcb.MaxValueNameLen, lpcb.MaxValueLen, lpcbSecurityDescriptor: PDWORD;
12620:                     lpftLastWriteTime: PFileTime): Longint; stdcall;
12621:     function RegQueryMultipleValues(hKey: HKEY; var ValList;
12622:         NumVals: DWORD; lpValueBuf: PKOOLChar; var ldwTotsize: DWORD): Longint; stdcall;
12623:     function RegQueryValue(hKey: HKEY; lpSubKey: PKOOLChar;
12624:         lpValue: PKOOLChar; var lpcbValue: Longint): Longint; stdcall;
12625:     function RegQueryValueEx(hKey: HKEY; lpValueName: PKOOLChar;
12626:         lpReserved: Pointer; lpType: PDWORD; lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12627:     function RegReplaceKey(hKey: HKEY; lpSubKey: PKOOLChar;
12628:         lpNewFile: PKOOLChar; lpOldFile: PKOOLChar): Longint; stdcall;
12629:     function RegRestoreKey(hKey: HKEY; lpFile: PKOOLChar; dwFlags: DWORD): Longint; stdcall;
12630:     function RegSaveKey(hKey: HKEY; lpFile: PKOOLChar;
12631:         lpSecurityAttributes: PSecurityAttributes): Longint; stdcall;
12632:     function RegSetValue(hKey: HKEY; lpSubKey: PKOOLChar;
12633:         dwType: DWORD; lpData: PKOOLChar; cbData: DWORD): Longint; stdcall;
12634:     function RegSetValueEx(hKey: HKEY; lpValueName: PKOOLChar;
12635:         Reserved: DWORD; dwType: DWORD; lpData: Pointer; cbData: DWORD): Longint; stdcall;
12636:     function RegUnloadKey(hKey: HKEY; lpSubKey: PKOOLChar): Longint; stdcall;
12637:     function RegisterEventSource(lpUNCServerName, lpSourceName: PKOOLChar): THandle; stdcall;
12638:     function ReportEvent(hEventLog: THandle; wType, wCategory: Word;
12639:         dwEventID: DWORD; lpUserSid: Pointer; wNumStrings: Word;
12640:             dwDataSize: DWORD; lpStrings, lpRawData: Pointer): BOOL; stdcall;
12641:     function SetFileSecurity(lpFileName: PKOOLChar; SecurityInformation: SECURITY_INFORMATION;
12642:         pSecurityDescriptor: PSecurityDescriptor): BOOL; stdcall;
12643:
12644:     Function wAddAtom( lpString : PKOOLchar ) : ATOM
12645:     Function wBeginUpdateResource( pFileName : PKOOLchar; bDeleteExistingResources : BOOL ) : THandle
12646: //Function wCallNamedPipe( lpNamedPipeName : PKOOLchar; lpInBuffer : Pointer; nInBufferSize : DWORD;
12647: lpOutBuffer : Pointer; nOutBufferSize : DWORD; var lpBytesRead : DWORD; nTimeOut : DWORD ) : BOOL

```

```

12647: //Function wCommConfigDialog( lpszName : PKOLChar; hWnd : HWND; var lpCC : TCommConfig) : BOOL
12648: Function wCompareString( Locale : LCID; dwCmpFlags : DWORD; lpString1 : PKOLChar; cchCount1 : Integer;
lpString2 : PKOLChar; cchCount2 : Integer) : Integer
12649: Function wCopyFile( lpExistingFileName, lpNewFileName : PKOLChar; bFailIfExists : BOOL) : BOOL
12650: //Function wCopyFileEx( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
TFNProgressRoutine; lpData : Pointer; pbCancel : PBool; dwCopyFlags : DWORD) : BOOL
12651: Function wCreateDirectory( lpPathName : PKOLChar; lpSecurityAttributes : PSecurityAttributes) : BOOL
12652: Function wCreateDirectoryEx(lpTemplateDirectory,
lpNewDirectory:PKOLChar;lpSecAttrib:PSecurityAttributes):BOOL;
12653: Function wCreateEvent( lpEventAttributes:PSecurityAttribs:bManualReset,
bInitialState:BOOL;lpName:PKOLChar):THandle;
12654: Function wCreateFile( lpFileName : PKOLChar; dwDesiredAccess, dwShareMode : DWORD; lpSecurityAttributes :
PSecurityAttributes; dwCreationDisposition, dwFlagsAndAttributes:DWORD:hTemplateFile:THandle):THandle
12655: Function wCreateFileMapping( hFile : THandle; lpFileMappingAttributes : PSecurityAttributes; flProtect,
dwMaximumSizeHigh, dwMaximumSizeLow : DWORD; lpName : PKOLChar) : THandle
12656: Function wCreateHardLink(lpFileName,
lpExistingFileName:PKOLChar;lpSecurityAttributes:PSecurityAttributes):BOOL
12657: Function CreateMailslot(lpName:PKOLChar;MaxMessSize:DWORD;lReadTimeout:DWORD;lpSecurityAttrib:PSecurityAttributes):THandle;
12658: Function wCreateNamedPipe( lpName : PKOLChar; dwOpenMode, dwPipeMode, nMaxInstances, nOutBufferSize,
nInBufferSize, nDefaultTimeOut : DWORD; lpSecurityAttributes : PSecurityAttributes) : THandle
12659: //Function CreateProcess( lpApplicationName : PKOLChar; lpCommandLine : PKOLChar; lpProcessAttributes,
lpThreadAttributes : PSecurityAttributes; bInheritHandles : BOOL; dwCreationFlags : DWORD; lpEnvironment :
Pointer;lpCurrentDirectory:PKOLChar;const lpStartupInfo:TStartupInfo;var
lpProcessInfo:TProcessInformation):BOOL
12660: Function wCreateSemaphore(lpSemaphoreAttributes: PSecurityAttributes; lInitialCount, lMaximumCount :
Longint; lpName : PKOLChar) : THandle
12661: Function wCreateWaitableTimer(lpTimerAttrbs:PSecurityAttribs;bManualReset:BOOL;lpTimerName:PKOLChar):THandle;
12662: Function wDefineDosDevice( dwFlags : DWORD; lpDeviceName, lpTargetPath : PKOLChar) : BOOL
12663: Function wDeleteFile( lpFileName : PKOLChar) : BOOL
12664: Function wEndUpdateResource( hUpdate : THandle; fDiscard : BOOL) : BOOL
12665: //Function
wEnumCalendarInfo(lpCalInfEnumProc:TFNCalInfEnumProc;Locale:LCID;Calendar:CALID;CalType:CALTYPE):BOOL,
12666: //Function wEnumDateFormats(lpDateFmtEnumProc: TFNDateFmtEnumProc; Locale : LCID; dwFlags : DWORD) : BOOL
12667: //Function
wEnumResourceNames(hModule:HMODULE;lpType:PKOLChar;lpEnumFunc:ENUMRESNAMEPROC;lParam:Longint):BOOL,
12668: //Function wEnumResourceTypes( hModule:HMODULE; lpEnumFunc:ENUMRESTYPEPROC;lParam:Longint):BOOL,
12669: //Function wEnumSystemCodePages( lpCodePageEnumProc : TFNCodepageEnumProc; dwFlags : DWORD) : BOOL
12670: //Function wEnumSystemLocales( lpLocaleEnumProc : TFNLocaleEnumProc; dwFlags : DWORD) : BOOL
12671: //Function wEnumTimeFormats(lpTimeFmtEnumProc:TFNTimeFmtEnumProc;Locale:LCID;dwFlags:DWORD):BOOL,
12672: Function wExpandEnvironmentStrings( lpSrc : PKOLChar; lpDst : PKOLChar; nSize : DWORD) : DWORD
12673: Procedure wFatalAppExit( uAction : UINT; lpMessageText : PKOLChar)
12674: //Function wFillConsoleOutputCharacter( hConsoleOutput : THandle; cCharacter : KOLChar; nLength : DWORD;
dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12675: Function wFindAtom( lpString : PKOLChar) : ATOM
12676: Function
wFindFirstChangeNotification(lpPathName:PKOLChar;bWatchSubtree:BOOL;dwNotifyFilter:DWORD):THandle;
12677: Function wFindFirstFile( lpFileName : PKOLChar; var lpFindFileData : TWIN32FindData) : THandle
12678: //Function wFindFirstFileEx( lpFileName : PKOLChar; fInfoLevelId : TFIndexInfoLevels; lpFindFileData :
Pointer; fSearchOp : TFIndexSearchOps; lpSearchFilter : Pointer; dwAdditionalFlags : DWORD) : BOOL
12679: Function wFindNextFile( hFindFile : THandle; var lpFindFileData : TWIN32FindData) : BOOL
12680: Function wFindResource( hModule : HMODULE; lpName, lpType : PKOLChar) : HRSRC
12681: Function wFindResourceEx( hModule : HMODULE; lpType, lpName : PKOLChar; wLanguage : Word) : HRSRC
12682: Function
wFoldString(dwMapFlags:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Integer):Integer;
12683: //Function wFormatMessage( dwFlags : DWORD; lpSource : Pointer; dwMessageId : DWORD; dwLanguageId :
DWORD; lpBuffer : PKOLChar; nSize : DWORD; Arguments : Pointer) : DWORD
12684: Function wFreeEnvironmentStrings( EnvBlock : PKOLChar) : BOOL
12685: Function wGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12686: Function wGetBinaryType( lpApplicationName : PKOLChar; var lpBinaryType : DWORD) : BOOL
12687: Function wGetCommandLine : PKOLChar
12688: //Function wGetCompressedFileSize( lpFileName : PKOLChar; lpFileSizeHigh : PDWORD) : DWORD
12689: Function wGetComputerName( lpBuffer : PKOLChar; var nSize : DWORD) : BOOL
12690: Function wGetConsoleTitle( lpConsoleTitle : PKOLChar; nSize : DWORD) : DWORD
12691: //Function wGetCurrencyFormat( Locale : LCID; dwFlags : DWORD; lpValue : PKOLChar; lpFormat :
PCurrencyFmt; lpCurrencyStr : PKOLChar; cchCurrency : Integer) : Integer
12692: Function wGetCurrentDirectory( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12693: //Function wGetDateFormat( Locale : LCID; dwFlags : DWORD; lpDate : PSystemTime; lpFormat : PKOLChar,
lpDateStr : PKOLChar; cchDate : Integer) : Integer
12694: //Function wGetDefaultCommConfig( lpszName:PKOLChar;var lpCC : TCommConfig;var lpdwSize:DWORD):BOOL
12695: Function wGetDiskFreeSpace( lpRootPathName : PKOLChar; var lpSectorsPerCluster, lpBytesPerSector,
lpNumberofFreeClusters, lpTotalNumberOfClusters : DWORD) : BOOL
12696: //Function wGetDiskFreeSpaceEx( lpDirectoryName : PKOLChar; var lpFreeBytesAvailableToCaller,
lpTotalNumberOfBytes, lpTotalNumberOfFreeBytes : PLargeInteger) : BOOL
12697: Function wGetDriveType( lpRootPathName : PKOLChar) : UINT
12698: Function wGetEnvironmentStrings : PKOLChar
12699: Function wGetEnvironmentVariable( lpName : PKOLChar; lpBuffer : PKOLChar; nSize : DWORD) : DWORD;
12700: Function wGetFileAttributes( lpFileName : PKOLChar) : DWORD
12701: //Function
wGetFileAttributesEx(lpFileName:PKOLChar;fInfoLevelId:TGetFileExInfoLevs;lpFileInform:Pointer):BOOL;
12702: Function wGetFullPathName(lpFileName:PKOLChar;nBufferLeng:WORD;lpBuffer:PKOLChar;var
lpFilePart:PKOLChar):DWORD;
12703: //Function wGetLocaleInfo(Locale:LCID; LCTYPE:LCTYPE;lpLCData:PKOLChar;cchData:Integer): Integer
12704: Function wGetLogicalDriveStrings( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12705: Function wGetModuleFileName( hModule : HINST; lpFilename : PKOLChar; nSize : DWORD) : DWORD
12706: Function wGetModuleHandle( lpModuleName : PKOLChar) : HMODULE
12707: //Function wGetNamedPipeHandleState( hNamedPipe : THandle; lpState, lpCurInstances, lpMaxCollectionCount,
lpCollectDataTimeout : PDWORD; lpUserName : PKOLChar; nMaxUserNameSize : DWORD) : BOOL

```

```

12708: //Function wGetNumberFormat( Locale : LCID; dwFlags:DWORD; lpValue:PKOLChar; lpFormat:PNumberFmt;
12709: lpNumberStr : PKOLChar; cchNumber : Integer ) : Integer
12710: Function wGetPrivateProfileInt(lpAppName:PKOLChar; nDefault:Integer;lpFileName:PKOLChar):UINT;
12711: Function wGetPrivateProfileSection(lpAppName:PKOLChar;lpRetrStr:PKOLChar;nSize:DWORD;pFileName:PKOLChar):DWORD;
12712: Function wGetPrivateProfileSectionNames(lpszReturnBuffer:PKOLChar;nSize:DWORD;lpFileName:PKOLChar):DWORD;
12713: Function wGetPrivateProfileString( lpAppName , lpKeyName , lpDefault : PKOLChar;lpReturnedStr : PKOLChar;
nSize:DWORD;lpFileName : PKOLChar) : DWORD
12714: Function wGetProfileInt( lpAppName , lpKeyName : PKOLChar; nDefault : Integer ) : UINT
12715: Function wGetProfileSection( lpAppName : PKOLChar; lpReturnedString : PKOLChar; nSize : DWORD ) : DWORD
12716: Function wGetProfileString(lpAppName,lpKeyName,
lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD):DWORD
12717: //Procedure wGetStartupInfo( var lpStartupInfo : TStartupInfo)
12718: // Function wGetStringTypeEx(Locale:LCID; dwInfoType:DWORD;lpSrcStr:PKOLChar;cchSrc:Integer,var
12719: lpCharType):BOOL
12720: Function wGetSystemDirectory( lpBuffer : PKOLChar; uSize : UINT ) : UINT
12721: Function wGetTempFileName( lpPathName , lpPrefixString: PKOLChar;uUnique:UINT;lpTempFileName:PKOLChar):UINT
12722: Function wGetTempPath( nBufferLength : DWORD; lpBuffer : PKOLChar ) : DWORD
12723: //Function wGetTimeFormat(Loc:LCID;dwFlgs:DWORD;lpTime:PSystemTime;lpFrm:PKOLChar;lpTimeStr:PKOLChar;cTime:Int):Int
12724: //Function wGetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
12725: //Function GetVolumeInformation( lpRootPathName : PKOLChar; lpVolumeNameBuffer : PKOLChar; nVolumeNameSize
: DWORD; lpVolumeSerialNumber : PDWORD; var lpMaximumComponentLength, lpFileSystemFlags : DWORD;
lpFileSystemNameBuffer : PKOLChar; nFileSystemNameSize : DWORD) : BOOL
12726: Function wGetWindowsDirectory( lpBuffer : PKOLChar; uSize : UINT ) : UINT
12727: Function wGlobalAddAtom( lpString : PKOLChar ) : ATOM
12728: Function wGlobalFindAtom( lpString : PKOLChar ) : ATOM
12729: Function wGlobalGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer ) : UINT
12730: Function wIsBadStringPtr( lpsz : PKOLChar; ucchMax : UINT ) : BOOL
12731: Function wLCMapString(Loc:LCID;dwMapFlgs:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Int):Int;
12732: Function wLoadLibrary( lpLibFileName : PKOLChar ) : HMODULE
12733: Function wLoadLibraryEx( lpLibFileName : PKOLChar; hFile : THandle; dwFlags : DWORD ) : HMODULE
12734: Function wMoveFile( lpExistingFileName, lpNewFileName : PKOLChar ) : BOOL
12735: Function wMoveFileEx( lpExistingFileName, lpNewFileName : PKOLChar; dwFlags : DWORD ) : BOOL
12736: //Function wMoveFileWithProgress( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
TFNProgressRoutine; lpData : Pointer; dwFlags : DWORD ) : BOOL
12737: Function wOpenEvent( dwDesiredAccess : DWORD; bInheritHandle : BOOL;lpName:PKOLChar ) : THandle
12738: Function wOpenFileMapping( dwDesiredAccess : DWORD; bInheritHandle:BOOL;lpName : PKOLChar ):THandle
12739: Function wOpenMutex( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar ):THandle
12740: Function wOpenWaitableTimer(dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpTimerName:PKOLChar):THandle
12741: Procedure wOutputDebugString( lpOutputString : PKOLChar )
12742: //Function wPeekConsoleInput(hConsoleInput:THandle;var lpBuffer:TInputRecord;nLength:DWORD;var
12743: lpNumberOfEventsRead:DWORD):BOOL;
12744: Function wQueryDosDevice( lpDeviceName : PKOLChar; lpTargetPath : PKOLChar; ucchMax : DWORD ) : DWORD
12745: //Function wQueryRecoveryAgents(p1:PKOLChar;var p2:Pointer;var p3:TRecoveryAgentInformation):DWORD
12746: //Function wReadConsole( hConsoleInput : THandle; lpBuffer : Pointer; nNumberOfCharsToRead : DWORD; var
12747: lpNumberOfCharsRead : DWORD; lpReserved : Pointer ) : BOOL
12748: //Function wReadConsoleInput(hConsInp:THandle;var lpBuf:TInpRec;nLength:DWORD;var
12749: lpNumbOfEventsRead:DWORD):BOOL;
12750: //Function wReadConsoleOutput( hConsoleOutput : THandle; lpBuffer : Pointer; dwBufferSize, dwBufferCoord
: TCoord; var lpReadRegion : TSmallRect ) : BOOL
12751: //Function wReadConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
DWORD; dwReadCoord : TCoord; var lpNumberOfCharsRead : DWORD ) : BOOL
12752: Function wRemoveDirectory( lpPathName : PKOLChar ) : BOOL
12753: //Function wScrollConsoleScreenBuffer( hConsoleOutput : THandle; const lpScrollRectangle : TSmallRect;
12754: lpClipRectangle : PSmallRect; dwDestinationOrigin : TCoord; var lpFill : TCharInfo ) : BOOL
12755: Function wSearchPath( lpPath,lpFileName,lpExtension:PKOLChar;nBufferLength:DWORD;lpBuffer:PKOLChar;var
12756: lpFilePart:PKOLChar):DWORD;
12757: Function wSetComputerName( lpComputerName : PKOLChar ) : BOOL
12758: Function wSetConsoleTitle( lpConsoleTitle : PKOLChar ) : BOOL
12759: Function wSetCurrentDirectory( lpPathName : PKOLChar ) : BOOL
12760: //Function wSetDefaultCommConfig( lpszName : PKOLChar; lpCC : PCommConfig; dwSize : DWORD) : BOOL
12761: Function wSetEnvironmentVariable( lpName, lpValue : PKOLChar ) : BOOL
12762: Function wSetFileAttributes( lpFileName : PKOLChar; dwFileAttributes : DWORD ) : BOOL
12763: //Function wSetLocaleInfo( Locale : LCID; LCTYPE : LCTYPE; lpLCData : PKOLChar ) : BOOL
12764: Function wSetVolumeLabel( lpRootPathName : PKOLChar; lpVolumeName : PKOLChar ) : BOOL
12765: //Function wUpdateResource(hUpdate:THandle;lpType,
12766: lpName:PKOLChar;wLanguage:Word;lpData:Ptr;cbData:DWORD):BOOL
12767: Function wVerLanguageName( wLang : DWORD; szLang : PKOLChar; nSize : DWORD ) : DWORD
12768: Function wWaitNamedPipe( lpNamedPipeName : PKOLChar; nTimeOut : DWORD ) : BOOL
12769: //Function wWriteConsole( hConsoleOutput : THandle; const lpBuffer : Pointer; nNumberOfCharsToWrite :
12770: DWORd; var lpNumberOfCharsWritten : DWORD; lpReserved : Pointer ) : BOOL
12771: //Function wWriteConsoleInput( hConsoleInput : THandle; const lpBuffer : TInputRecord; nLength : DWORD;
12772: var lpNumberOfEventsWritten : DWORD ) : BOOL
12773: //Function wWriteConsoleOutput(hConsoleOutput:THandle; lpBuffer:Pointer; dwBufferSize,dwBufferCoord :
12774: TCoord; var lpWriteRegion : TSmallRect ) : BOOL
12775: //Function wWriteConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
12776: DWORD; dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD ) : BOOL
12777: Function wWritePrivateProfileSection( lpAppName, lpString, lpFileName : PKOLChar ) : BOOL
12778: Function wWritePrivateProfileString( lpAppName, lpKeyName, lpString, lpFileName : PKOLChar ) : BOOL
12779: Function wWriteProfileSection( lpAppName, lpString : PKOLChar ) : BOOL
12780: Function wWriteProfileString( lpAppName, lpKeyName, lpString : PKOLChar ) : BOOL
12781: Function wlstrcat( lpString1, lpString2 : PKOLChar ) : PKOLChar
12782: Function wlstrcmp( lpString1, lpString2 : PKOLChar ) : Integer
12783: Function wlstrcmpi( lpString1, lpString2 : PKOLChar ) : Integer
12784: Function wlstrcpy( lpString1, lpString2 : PKOLChar ) : PKOLChar

```

```

12775: Function wlstrcpyn( lpString1, lpString2 : PKOLChar; iMaxLength : Integer ) : PKOLChar
12776: Function wlstrlen( lpString : PKOLChar ) : Integer
12777: //Function wMultinetGetConnectionPerformance( lpNetResource : PNetResource; lpNetConnectInfoStruct : PNetConnectInfoStruct ) : DWORD
12778: //Function wWNetAddConnection2(var lpNetResource:TNetResource;lpPassw,
12779: lpUserName:PKOLChar;dwFlags:DWORD):DWORD;
12780: //Function wWNetAddConnection3( hwndOwner : HWND; var lpNetResource:TNetResource;lpPassword,
12781: lpUserName:PKOLChar; dwFlags : DWORD ) : DWORD
12782: Function wWNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PKOLChar ) : DWORD
12783: Function wWNetCancelConnection2( lpName : PKOLChar; dwFlags : DWORD; fForce : BOOL ) : DWORD
12784: Function wWNetCancelConnection( lpName : PKOLChar; fForce : BOOL ) : DWORD
12785: //Function wWNetConnectionDialog1( var lpConnDlgStruct : TConnectDlgStruct ) : DWORD
12786: //Function wWNetDisconnectDialog1( var lpConnDlgStruct : TDIsDlgStruct ) : DWORD
12787: Function wWNetGetConnection(lpLocalName:PKOLChar;lpRemoteName:PKOLChar; var lpnLength:DWORD):DWORD;
12788: Function wWNetGetLastError( var lpError : DWORD; lpErrorBuf : PKOLChar; nErrorBufSize : DWORD; lpNameBuf
12789: : PKOLChar; nNameBufSize : DWORD ) : DWORD
12790: //Function wWNetGetNetworkInformation(lpProvider:PKOLChar;var lpNetInfoStruct:TNetInfoStruct):DWORD;
12791: Function wWNetGetProviderName(dwNetType:DWORD;lpProviderName:PKOLChar;var lpBufferSize:DWORD):DWORD;
12792: //Function wWNetGetResourceParent(lpNetResource:PNetResource;lpBuffer:Pointer;var cbBuffer:DWORD):DWORD;
12793: //Function wWNetGetUniversalName(lpLocalPath:PKOLChar;dwInfoLevel:DWORD;lpBuffer:Ptr;var
12794: lpBufferSize:DWORD):DWORD;
12795: Function wWNet GetUser( lpName : PKOLChar; lpUserName : PKOLChar; var lpnLength : DWORD ) : DWORD
12796: // Function wWNetOpenEnum(dwScope,dwType,dwUsage:DWORD;lpNetResource:PNetRes;var lphEnum:THandle):DWORD;
12797: // Function wWNetSetConnection( lpName : PKOLChar; dwProperties : DWORD; pvValues : Pointer ) : DWORD
12798: //Function wWNetUseConnection(hwndOwner:HWND;var
12799: lpNetResource:TNetResource;lpUserID:PKOLChar;lpPassword:PKOLChar; dwFlags:DWORD;lpAccessName:PKOLChar;var
12800: lpBufferSize:DWORD;var lpResult:DWORD):DWORD
12801: Function wGetFileVersionInfo(lptstrFilename:PKOLChar;dwHandle,dwLen:DWORD;lpData:Pointer):BOOL;
12802: Function wGetFileVersionInfoSize( lptstrFilename : PKOLChar; var lpdwHandle : DWORD ) : DWORD
12803: Function wVerFindfile( uFlags : DWORD; szFileName, szWinDir, szAppDir, szCurDir : PKOLChar; var
12804: lpuCurDirLen : UINT; szDestDir : PKOLChar; var lpuDestDirLen : UINT ) : DWORD
12805: Function wVerInstallFile( uFlags : DWORD; szSrcFileName, szDestFileName, szSrcDir, szDestDir, szCurDir,
12806: szTmpFile : PKOLChar; var lpuTmpFileLen : UINT ) : DWORD
12807: //Function wVerQueryValue(pBlock:Pointer;lpSubBlock:PKOLChar;var lplpBuffer:Ptr;var puLen:UINT):BOOL;
12808: //Function wGetPrivateProfileStruct(lpszSection,
12809: lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12810: //Function wWritePrivateProfileStruct(lpszSection,
12811: lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12812: Function wAddFontResource( FileName : PKOLChar ) : Integer
12813: //Function wAddFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector ) : Integer
12814: Function wCopyEnhMetaFile( p1 : HENHMETAFILE; p2 : PKOLChar ) : HENHMETAFILE
12815: Function wCopyMetaFile( p1 : HMETAFILE; p2 : PKOLChar ) : HMETAFILE
12816: //Function wCreateColorSpace( var ColorSpace : TLogColorSpace ) : HCOLORSPACE
12817: //Function wCreateDC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpDvmInit : PDeviceMode ) : HDC
12818: // Function wCreateEnhMetaFile( DC : HDC; FileName : PKOLChar; Rect : PRect; Desc : PKOLChar ) : HDC
12819: Function wCreateFont( nHeight, nWidth, nEscapement, nOrientation, fnWeight : Integer; fdwItalic,
12820: fdwUnderline, fdwStrikeOut, fdwCharSet, fdwOutputPrec, fdwClipPrecision, fdwQualy,
12821: fdwPitchAndFamily:DWORD;lpszFace:PKOLChar):HFONT;
12822: Function wCreateFontIndirect( const p1 : TLogFont ) : HFONT
12823: //Function wCreateFontIndirectEx( const p1 : PEnumLogFontExDV ) : HFONT
12824: // Function wCreateIC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpDvmInit : PDeviceMode ) : HDC
12825: Function wCreateMetaFile( p1 : PKOLChar ) : HDC
12826: Function wCreateScalableFontResource( p1 : DWORD; p2, p3, p4 : PKOLChar ) : BOOL
12827: //Function wDeviceCapabilities(pDriverNa,pDeviceNam,
12828: pPort:PKOLChar;iIdx:Int;pOut:PKOLChar;DevMod:PDeviceMode):Int;
12829: // Function wEnumFontFamilies( DC : HDC; p2 : PKOLChar; p3 : TFontEnumProc; p4 : LPARAM ) : BOOL
12830: //Function wEnumFontFamiliesEx(DC:HDC;var p2:TLogFont;p3:TFontEnumProc;p4:LPARAM;p5:DWORD):BOOL;
12831: //Function wEnumFonts(DC:HDC;lpszFace:PKOLChar;fntermprc:TFontEnumProc;lpszData:PKOLChar):Integer;
12832: //Function wEnumICMPProfiles( DC : HDC; ICMPProc : TFNICKMEnumProc; p3 : LPARAM ) : Integer
12833: //Function wExtTextOut(DC:HDC;X,
12834: Y:Int;Options:Longint;Rect:PRect;Str:PKOLChar;Count:Longint;Dx:PInteger:BOOL
12835: //Function wGetCharABCWidths( DC : HDC; FirstChar, LastChar : UINT; const ABCStructs ) : BOOL
12836: //Function wGetCharABCWidthsFloat( DC : HDC; FirstChar, LastChar : UINT; const ABCFloatSturcts ) : BOOL
12837: //Function wGetCharWidth32( DC : HDC; FirstChar, LastChar : UINT; const Widths ) : BOOL
12838: //Function wGetCharWidth( DC : HDC; FirstChar, LastChar : UINT; const Widths ) : BOOL
12839: // Function wGetCharWidthFloat( DC : HDC; FirstChar, LastChar : UINT; const Widths ) : BOOL
12840: // Function wGetCharacterPlacement(DC:HDC;p2:PKOLChar;p3,p4:BOOL;var p5:TGCPResults;p6:DWORD):DWORD
12841: Function wGetEnhMetaFile( p1 : PKOLChar ) : HENHMETAFILE
12842: Function wGetEnhMetaFileDescription( p1 : HENHMETAFILE; p2 : UINT; p3 : PKOLChar ) : UINT
12843: // Function wGetGlyphIndices( DC : HDC; p2 : PKOLChar; p3 : Integer; p4 : PWORD; p5 : DWORD ) : DWORD
12844: // Function wGetGlyphOutline( DC : HDC; uChar, uFormat:UINT;const lpgm:TGlyphMetrics; cbBuffer : DWORD,
12845: lpvBuffer : Pointer; const lpmat2 : TMat2 ) : DWORD
12846: Function wGetICMPProfile( DC : HDC; var Size : DWORD; Name : PKOLChar ) : BOOL
12847: //Function wGetLogColorSpace( p1 : HCOLORSPACE; var ColorSpace : TLogColorSpace; Size : DWORD ) : BOOL
12848: Function wGetMetaFile( p1 : PKOLChar ) : HMETAFILE
12849: Function wGetObject( p1 : HGDIOBJ; p2 : Integer; p3 : Pointer ) : Integer
12850: //Function wGetOutlineTextMetrics( DC : HDC; p2 : UINT; OTMetricStructs : Pointer ) : UINT
12851: //Function wGetTextExtentExPoint(DC:HDC;p2:PKOLChar; p3,p4:Integer;p5,p6:PInteger;var p7:TSize):BOOL
12852: Function wGetTextExtentPoint32( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize ) : BOOL
12853: Function wGetTextExtentPoint( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize ) : BOOL
12854: Function wGetTextFace( DC : HDC; Count : Integer; Buffer : PKOLChar ) : Integer
12855: //Function wGetTextMetrics( DC : HDC; var TM : TTTextMetric ) : BOOL
12856: Function wPolyTextOut( DC : HDC; const PolyTextArray, Strings : Integer ) : BOOL
12857: Function wRemoveFontResource( FileName : PKOLChar ) : BOOL
12858: //Function wRemoveFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector ) : BOOL
12859: //Function wResetDC( DC : HDC; const InitData : TDeviceMode ) : HDC
12860: Function wSetICMPProfile( DC : HDC; Name : PKOLChar ) : BOOL
12861: //Function wStartDoc( DC : HDC; const p2 : TDocInfo ) : Integer

```

```

12848: Function wTextOut( DC : HDC; X, Y : Integer; Str : PKOLChar; Count : Integer) : BOOL
12849: Function wUpdateICMRegKey( p1 : DWORD; p2, p3 : PKOLChar; p4 : UINT) : BOOL
12850: Function wwglUseFontBitmaps( DC : HDC; p2, p3, p4 : DWORD) : BOOL
12851: //Function wwglUseFontOutlines(p1:HDC; p2,p3,p4:DWORD;p5,p6:Single;p7:Int;p8:PGlyphMetricsFloat):BOOL
12852: Function wAppendMenu( hMenu : HMENU; uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12853: Function wCallMsgFilter( var lpMsg : TMsg; nCode : Integer) : BOOL
12854: //Function
12855: wCallWindowProc(lpPrevWndFunc:TFNWndProc;hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT
12856: //Function wChangeDisplaySettings( var lpDevMode : TDeviceMode; dwFlags : DWORD) : Longint
12857: // Function wChangeDisplaySettingsEx( lpszDeviceName:PKOLChar;var lpDevMode: TDeviceMode; wnd : HWND;
12858: dwFlags : DWORD; lParam : Pointer) : Longint
12859: Function wChangeMenu(hMenu:HMENU;cmd:UINT;lpszNewItem:PKOLChar;cmdInsert:UINT;flags:UINT):BOOL
12860: Function wCharLower( lpsz : PKOLChar) : PKOLChar
12861: Function wCharLowerBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12862: Function wCharNext( lpsz : PKOLChar) : PKOLChar
12863: // Function wCharNextEx( CodePage : Word; lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12864: Function wCharToOem( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
12865: Function wCharToOemBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
12866: Function wCharUpper( lpsz : PKOLChar) : PKOLChar
12867: Function wCharUpperBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12868: Function wCopyAcceleratorTable( hAccelSrc : HACCEL; var lpAccelDst, cAccelEntries : Integer) : Integer
12869: Function wCreateAcceleratorTable( var Accel, Count : Integer) : HACCEL
12870: //Function wCreateDesktop(lpszDesktop,
12871: lpszDevice:PKOLChar;pDevMode:DDeviceMode;dwFlags:DWORD;dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HDESK
12872: //Function wCreateDialogIndirectParam( hInstance : HINST; const lpTemplate : TDlgTemplate; hWndParent :
12873: HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
12874: //Function wCreateDialogParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND;
12875: lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
12876: Function wCreateMDIWindow( lpClassName, lpWindowName:PKOLChar;dwStyle: DWORD; X,Y,nWidth,nHeight:Integer;
12877: hWndParent : HWND; hInstance : HINST; lParam : LPARAM) : HWND
12878: //Function wCreateWindowEx(dwExStyle:DWORD;lpClassName:PKOLChar;lpWindowName:PKOLChar;dwStyle WORD;X,Y,
12879: nWidth, nHeight:Int WndParent:HWND;hMenu:HMENU;hInstance:HINST;lpParam:Pointer):HWND
12880: //Function wCreateWindowStation(lpinsta:PKOLChar;dwReserv,
12881: dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HWINSTA;
12882: Function wDefDlgProc( hDlg : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
12883: Function wDefFrameProc(hWnd,hWndMDIClient:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
12884: Function wDefMDIChildProc( hWnd : HWND; uMsg : UINT; wParam : WPARAM; lParam: LPARAM):LRESULT;
12885: Function wDefWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam: LPARAM) : LRESULT
12886: //Function wDialogBoxIndirectParam( hInstance : HINST; const lpDialogTemplate : TDlgTemplate; hWndParent
12887: : HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : Integer
12888: //Function wDialogBoxParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND; lpDialogFunc
12889: : TFNDlgProc; dwInitParam : LPARAM) : Integer
12890: Function wDispatchMessage( const lpMsg : TMsg) : Longint
12891: Function wDlgDirList(hDlg:HWND;lpPathSpec:PKOLChar;nIDListBox,
12892: nIDStaticPath:Integer;ufileType:UINT):Integer;
12893: Function wDlgDirListComboBox(hDlg:HWND;lpPathSpec:PKOLChar;nIDComboBox,
12894: nIDStaticPath:Integer;ufiletype:UINT):Int;
12895: Function wDlgDirSelectComboBoxEx( hDlg : HWND; lpString:PKOLChar;nCount,nIDComboBox:Integer): BOOL
12896: Function wDlgDirSelectEx( hDlg : HWND; lpString : PKOLChar; nCount, nIDListBox : Integer) : BOOL
12897: //Function wDrawState(dc:HDC;Brush:HBRUSH,CBFunc:TFNDrawStateProc;lData:LPARA;wDat:WPARA;x,y,cx,
12898: cy:Int;Flags:UINT):BOOL;
12899: Function wDrawText(hdc:HDC;lpString:PKOLChar;nCount:Integer;var lpRect:TRect;uFormat:UINT):Integer;
12900: Function wFindWindow( lpClassName, lpWindowName : PKOLChar) : HWND
12901: Function wFindWindowEx( Parent, Child : HWND; ClassName, WindowName : PKOLChar) : HWND
12902: //Function wGetAltTabInfo(hwnd:HWND;iItem:Int;var
12903: pati:TAltTabInfo;pszItemText:PKOLChar;cchItemText:UINT):BOOL;
12904: // Function wGetClassInfo( hInstance : HINST; lpClassName : PKOLChar; var lpWndClass : TWndClass) : BOOL
12905: //Function wGetClassInfoEx( Instance : HINST; Classname : PKOLChar; var WndClass : TWndClassEx) : BOOL
12906: Function wGetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
12907: Function wGetClipboardFormatName( format : UINT; lpszFormatName:PKOLChar;cchMaxCount:Integer):Integer;
12908: Function wGetDlgItemText( hDlg: HWND;nIDDlgItem:Integer;lpString:PKOLChar;nMaxCount:Integer):UINT
12909: Function wGetKeyNameText( lParam : Longint; lpString : PKOLChar; nSize : Integer) : Integer
12910: Function wGetKeyboardLayoutName( pwszKLID : PKOLChar) : BOOL
12911: //Function wGetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; var p4 : TMenuItemInfo) : BOOL
12912: Function wGetMenuString(hMenu:HMENU;uIDItem:UINT;lpString:PKOLChar;nMaxCount:Integer;uFlag:UINT):Integer;
12913: Function wGetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
12914: //Function wGetProp( hWnd : HWND; lpString : PKOLChar) : THandle
12915: //Function wGetTabbedTextExtent( hdc : HDC; lpString : PKOLChar; nCount, nTabPositions : Integer; var
12916: lpnTabStopPositions ) : DWORD
12917: //Function wGetObjectInform(hObj:THandle;nIndex:Int;pvInfo:Ptr;nLength:DWORD;var
12918: lpnLengthNeed:DWORD):BOOL;
12919: Function wGetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
12920: Function wGetWindowModuleFileName( hwnd : HWND; pszFileName : PKOLChar; cchFileNameMax : UINT) : UINT
12921: Function wGetWindowText( hWnd : HWND; lpString : PKOLChar; nMaxCount : Integer) : Integer
12922: Function wGetWindowTextLength( hWnd : HWND) : Integer
12923: //Function wGrayString(hDC:HDC;hBrush:HBRUSH;lpOutFunc:TFNGrayStrProc;lpDat:LPARA;nCnt,X,Y,nWidt,
12924: nHeight:Int):BOOL;
12925: Function wInsertMenu( hMenu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12926: //Function wInsertMenuItem( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
12927: Function wIsCharAlpha( ch : KOLChar) : BOOL
12928: Function wIsCharAlphaNumeric( ch : KOLChar) : BOOL
12929: Function wIsCharLower( ch : KOLChar) : BOOL
12930: Function wIsCharUpper( ch : KOLChar) : BOOL
12931: Function wIsDialogMessage( hDlg : HWND; var lpMsg : TMsg) : BOOL
12932: Function wLoadAccelerators( hInstance : HINST; lpTableName : PKOLChar) : HACCEL
12933: Function wLoadBitmap( hInstance : HINST; lpBitmapName : PKOLChar) : HBITMAP

```

```

12920: Function wLoadCursor( hInstance : HINST; lpCursorName : PKOLChar ) : HCURSOR
12921: Function wLoadCursorFromFile( lpFileName : PKOLChar ) : HCURSOR
12922: Function wLoadIcon( hInstance : HINST; lpIconName : PKOLChar ) : HICON
12923: Function wLoadImage(hInst:HINST;ImageName:PKOLChar;ImageType:UINT;X,Y:Integer;Flags:UINT) : THandle
12924: Function wLoadKeyboardLayout( pwszKLID : PKOLChar; Flags : UINT ) : HKL
12925: Function wLoadMenu( hInstance : HINST; lpMenuName : PKOLChar ) : HMENU
12926: //Function wLoadMenuIndirect( lpMenuTemplate : Pointer ) : HMENU
12927: Function wLoadString(hInst:HINST;uID: UINT; lpBuffer:PKOLChar;nBufferMax:Integer):Integer
12928: Function wMapVirtualKey( uCode, uMapType : UINT ) : UINT
12929: Function wMapVirtualKeyEx( uCode, uMapType : UINT; dwhk1 : HKL ) : UINT
12930: Function wMessageBox( hWnd : HWND; lpText, lpCaption : PKOLChar; uType : UINT ) : Integer
12931: Function wMessageBoxEx( hWnd:HWND; lpText,lpCaption:PKOLChar;uType:UINT;wLanguageId:Word) : Integer
12932: //Function wMessageBoxBoxIndirect( const MsgBoxParams : TMsgBoxParams ) : BOOL
12933: Function wModifyMenu( hMnu : HMENU; uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar ) : BOOL
12934: //Function woemToAnsi( const lpszSrc : LPCSTR; lpszDst : LPSTR ) : BOOL
12935: //Function woemToAnsiBuf( lpszSrc : LPCSTR; lpszDst : LPSTR; cchDstLength : DWORD ) : BOOL
12936: //Function woemToChar( lpszSrc : PKOLChar; lpszDst : PKOLChar ) : BOOL
12937: Function woemToCharBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD ) : BOOL
12938: Function wOpenDesktop(lpszDesktop:PKOLChar;dwFlags:DWORD;fInherit:BOOL;dwDesiredAccess:DWORD):HDESK
12939: Function wOpenWindowStation( lpszWinSta : PKOLChar; finherit : BOOL; dwDesiredAccess : DWORD ) : HWINSTA
12940: Function wPeekMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax, wRemoveMsg:UINT ):BOOL
12941: Function wPostMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
12942: Function wPostThreadMessage(idThread:DWORD;Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
12943: Function wRealGetWindowClass( hWnd : HWND; pszType : PKOLChar; cchType : UINT ) : UINT
12944: // Function wRegisterClass( const lpWndClass : TWndClass ) : ATOM
12945: // Function wRegisterClassEx( const WndClass : TWndClassEx ) : ATOM
12946: Function wRegisterClipboardFormat( lpszFormat : PKOLChar ) : UINT
12947: // Function wRegisterDeviceNotification(hRecipient:THandle;NotificFilter:Pointer;Flags:DWORD):HDEVNOTIFY
12948: Function wRegisterWindowMessage( lpString : PKOLChar ) : UINT
12949: Function wRemoveProp( hWnd : HWND; lpString : PKOLChar ) : THANDLE
12950: Function wSendDlgItemMessage(hDlg:HWND;nIDDlItem:Integer;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
12951: Function wSendMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT
12952: //Function wSendMessageCallback( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM;
12953: lpResultCallBack : TFNSendAsyncProc; dwData : DWORD ) : BOOL
12954: Function wSendTimeout(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM; fuFlags,uTimeout:UINT;var
12955: lpdwResult:DWORD) : LRESULT
12956: Function wSendNotifyMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
12957: Function wSetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : DWORD
12958: Function wSetDlgItemText( hDlg:HWND;nIDDlItem : Integer; lpString : PKOLChar ) : BOOL
12959: //Function wSetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo ) : BOOL
12960: Function wSetProp( hWnd : HWND; lpString : PKOLChar; hData : THANDLE ) : BOOL
12961: Function wSetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : Longint
12962: //Function wSetWindowsHook( nFilterType : Integer; pfnFilterProc : TFNHookProc ) : HHOOK
12963: //Function wSetWindowsHookEx(idHook:Integer;lpfn:TFNHookProc;hmod:HINST;dwThreadId:DWORD):HHOOK;
12964: // Function wSystemParametersInfo( uiAction, uiParam : UINT; pvParam : Pointer; fWinIni : UINT):BOOL
12965: Function wTabbedTextOut(hdc:HDC;X,Y:Int;lpString:PKOLChar;nCount,nTabPositions:Int;var
12966: lpnTabStopPositions,nTabOrigin:Int):Longint;
12967: Function wTranslateAccelerator( hWnd : HWND; hAccTable : HACCEL; var lpMsg : TMsg ) : Integer
12968: Function wUnregisterClass( lpClassName : PKOLChar; hInstance : HINST ) : BOOL
12969: Function wVKeyScanEx( ch : KOLChar; dwhkl : HKL ) : SHORT
12970: Function wWinHelp( hWndMain : HWND; lpszHelp : PKOLChar; uCommand : UINT; dwData : DWORD ) : BOOL
12971: Function wwsprintf( Output : PKOLChar; Format : PKOLChar ) : Integer
12972: Function wwwsprintf( Output : PKOLChar; Format : PKOLchar; arglist : va_list ) : Integer
12973:
12974: //TestDrive!
12975: 'SID_REVISION','LongInt'(1);'FILENAME_ADVAPI32','String').SetString('ADVAPI32.DLL
12976: 'PROC_CONVERTSIDTOSTRINGSIDA','String').SetString('ConvertSidToStringSidA'
12977: Function GetDomainUserSidS(const domainName:String;const userName:String; var foundDomain:String):String
12978: Function GetLocalUserSidStr( const UserName : string ) : string
12979: Function getPid4User( const domain : string; const user : string; var pid : dword ) : boolean
12980: Function Impersonate2User( const domain : string; const user : string ) : boolean
12981: Function GetProcessUserBypid( pid : DWORD; var UserName, Domain : AnsiString ) : Boolean
12982: Function KillProcessbyname( const exename : string; var found : integer ) : integer
12983: Function getWinProcessList : TStringList
12984: end;
12985:
12986: procedure SIRegister_AfSafeSync(CL: TPSPascalCompiler);
12987: begin
12988:   'AfMaxSyncSlots','LongInt'( 64 );
12989:   'AfSynchronizeTimeout','LongInt'( 2000 );
12990:   TafSyncSlotID', 'DWORD
12991:   TafSyncStatistics', 'record MessagesCount:Int;TimeoutMessages:Int;DisabledMessages:Int;end';
12992:   TafSafeSyncEvent', 'Procedure ( ID : TafSyncSlotID )
12993:   TafSafeDirectSyncEvent', 'Procedure
12994:   Function AfNewSyncSlot( const AEvent : TafSafeSyncEvent ) : TafSyncSlotID
12995:   Function AfReleaseSyncSlot( const ID : TafSyncSlotID ) : Boolean
12996:   Function AfEnableSyncSlot( const ID : TafSyncSlotID; Enable : Boolean ) : Boolean
12997:   Function AfValidateSyncSlot( const ID : TafSyncSlotID ) : Boolean
12998:   Function AfSyncEvent( const ID : TafSyncSlotID; Timeout : DWORD ) : Boolean
12999:   Function AfDirectSyncEvent( Event : TafSafeDirectSyncEvent; Timeout : DWORD ) : Boolean
13000:   Function AfIsSyncMethod : Boolean
13001:   Function AfSyncWnd : HWnd
13002:   Function AfSyncStatistics : TafSyncStatistics
13003:   Procedure AfClearSyncStatistics
13004: end;
13005:

```

```

13006: procedure SIRegister_AfComPortCore(CL: TPSCompiler);
13007: begin
13008:   'fBinary', 'LongWord')($00000001);
13009:   'fParity', 'LongWord')($00000002);
13010:   'fOutxCtsFlow', 'LongWord').SetUInt($00000004);
13011:   'fOutxDsrFlow', 'LongWord')($00000008);
13012:   'fDtrControl', 'LongWord')($00000030);
13013:   'fDtrControlDisable', 'LongWord')($00000000);
13014:   'fDtrControlEnable', 'LongWord')($00000010);
13015:   'fDtrControlHandshake', 'LongWord')($00000020);
13016:   'fDsSensitivity', 'LongWord')($00000040);
13017:   'fTxCContinueOnXoff', 'LongWord')($00000080);
13018:   'fOutX', 'LongWord')($00000100);
13019:   'fInX', 'LongWord')($00000200);
13020:   'fErrorChar', 'LongWord')($00000400);
13021:   'fNull', 'LongWord')($00000800);
13022:   'fRtsControl', 'LongWord')($00003000);
13023:   'fRtsControlDisable', 'LongWord')($00000000);
13024:   'fRtsControlEnable', 'LongWord')($00001000);
13025:   'fRtsControlHandshake', 'LongWord')($00002000);
13026:   'fRtsControlToggle', 'LongWord')($00003000);
13027:   'fAbortOnError', 'LongWord')($00004000);
13028:   'fDummy2', 'LongWord')($FFFF8000);
13029:   TAFCoreEvent', '(ceOutFree, ceLineEvent, ceNeedReadData, ceException)
13030:   FindClass('TOBJECT'), 'EAFComPortCoreError
13031:   FindClass('TOBJECT'), 'TAFComPortCore
13032:   TAFComPortCoreEvent', 'Procedure ( Sender : TAFComPortCore; Event'
13033:     +'tKind : TAFCoreEvent; Data : DWORD)
13034:   SIRegister_TAFComPortCoreThread(CL);
13035:   SIRegister_TAFComPortEventThread(CL);
13036:   SIRegister_TAFComPortWriteThread(CL);
13037:   SIRegister_TAFComPortCore(CL);
13038:   Function FormatDeviceName( PortNumber : Integer ) : string
13039: end;
13040:
13041: procedure SIRegister_ApplicationFileIO(CL: TPSCompiler);
13042: begin
13043:   TAFIOFileStreamEvent', 'Function ( const fileName : String; mode: Word ) : TStream
13044:   TAFIOFileExistsEvent', 'Function ( const fileName : String ) : Boolean
13045:   SIRegister_TApplicationFileIO(CL);
13046:   TDataFileCapability', '(dfcRead, dfcWrite)
13047:   TDataFileCapabilities', 'set of TDataFileCapability
13048:   SIRegister_TDataFile(CL);
13049:   //TDataFileClass', 'class of TDataFile
13050:   Function ApplicationFileIODefined : Boolean
13051:   Function CreateFileStream(const fileName: String; mode: WordfmShareDenyNone):TStream
13052:   Function FileStreamExists(const fileName: String) : Boolean
13053:   //Procedure Register
13054: end;
13055:
13056: procedure SIRegister_ALFBXLib(CL: TPSCompiler);
13057: begin
13058:   TALFBXFieldType', '(uftUnknown, uftNumeric, uftChar, uftVarchar'
13059:     +', uftCstring, uftSmallint, uftInteger, uftQuad, uftFloat, uftDoublePrecision'
13060:     +'on, uftTimestamp, uftBlob, uftBlobId, uftDate, uftTime, uftInt64, uftArray, uftNull )
13061:   TALFBXScale', 'Integer
13062:   FindClass('TOBJECT'), 'EALFBXConvertError
13063:   SIRegister_EALFBXError(CL);
13064:   SIRegister_EALFBXException(CL);
13065:   FindClass('TOBJECT'), 'EALFBXGFixError
13066:   FindClass('TOBJECT'), 'EALFBXDSQLError
13067:   FindClass('TOBJECT'), 'EALFBXDynError
13068:   FindClass('TOBJECT'), 'EALFBXBakError
13069:   FindClass('TOBJECT'), 'EALFBXGSecError
13070:   FindClass('TOBJECT'), 'EALFBXLicenseError
13071:   FindClass('TOBJECT'), 'EALFBXGStatError
13072:   //EALFBXExceptionClass', 'class of EALFBXError
13073:   TALFBXCharacterSet', '(csNONE, csASCII, csBIG_5, csCYRL, csDOS4'
13074:     +'37, csDOS850, csDOS852, csDOS857, csDOS860, csDOS861, csDOS863, csDOS865, '
13075:     +'csEUCL_0208, csGB_2312, csISO8859_1, csISO8859_2, csKSC_5601, csNEXT, csOC'
13076:     +'TET, csSJIS_0208, csUNICODE_FSS, csUTF8, csWIN1250, csWIN1251, csWIN1252, '
13077:     +'csWIN1253, csWIN1254, csDOS737, csDOS775, csDOS858, csDOS862, csDOS864, c'
13078:     +'sDOS866, csDOS869, csWIN1255, csWIN1256, csWIN1257, csISO8859_3, csISO8859'
13079:     +'4, csISO8859_5, csISO8859_6, csISO8859_7, csISO8859_8, csISO8859_9, csISO'
13080:     +'8859_13, csKOI8R, csWIN1258, csTIS620, csGBK, csCP943C )
13081:   TALFBXTransParam', '(tpConsistency, tpConcurrency, tpShared, tp'
13082:     +'Protected, tpExclusive, tpWait, tpNowait, tpRead, tpWrite, tpLockRead, tpL'
13083:     +'ockWrite, tpVerbTime, tpCommitTime, tpIgnoreLimbo, tpReadCommitted, tpAuto'
13084:     +'Commit, tpRecVersion, tpNoRecVersion, tpRestartRequests, tpNoAutoUndo, tpLockTimeout )
13085:   TALFBXTransParams', 'set of TALFBXTransParam
13086:   Function ALFBXStrToCharacterSet( const CharacterSet : AnsiString ) : TALFBXCharacterSet
13087:   Function ALFBXCreateDBParams( Params : AnsiString; Delimiter : Char ) : AnsiString
13088:   Function ALFBXCreateBlobParams( Params : AnsiString; Delimiter : Char ) : AnsiString
13089:   'cALFBXMaxParamLength', 'LongInt'( 125 );
13090:   TALFBXParamsFlag', '(pfNotInitialized, pfNotNullable )
13091:   TALFBXParamsFlags', 'set of TALFBXParamsFlag
13092:   //PALFBXSQLVar', '^TALFBXSQLVar // will not work
13093:   //PALFBXSQLDaData', '^TALFBXSQLDaData // will not work
13094:   TALFBXStatementType', '(stSelect, stInsert, stUpdate, stDelete, '

```

```

13095: +' stDDL, stGetSegment, stPutSegment, stExecProcedure, stStartTrans, stComm'
13096: +'t, stRollback, stSelectForUpdate, stSetGenerator, stSavePoint )
13097: SIRегистер_TALFBXSQLDA(CL);
13098: //PALFBXPtrArray', '^PALFBXPtrArray // will not work
13099: SIRегистер_TALFBXPoolStream(CL);
13100: //PALFBXBlobData', '^PALFBXBlobData // will not work
13101: TALFBXBlobData', 'record Size : Integer; Buffer : string; end
13102: //PALFBXArrayDesc', 'TALFBXArrayDesc // will not work
13103: //TALFBXArrayDesc', 'TISCArrayDesc
13104: //TALFBXBlobDesc', 'TISCBlobDesc
13105: //PALFBXArrayInfo', '^PALFBXArrayInfo // will not work
13106: //PALFBXArrayInfo', 'record index : Integer; size : integer; info: TALFBXArrayDesc; end
13107: SIRегистер_TALFBXSQLResult(CL);
13108: //TALFBXSQLResultClass', 'class of TALFBXSQLResult
13109: SIRегистер_TALFBXSQLParams(CL);
13110: //TALFBXSQLParamsClass', 'class of TALFBXSQLParams
13111: TALFBXDSQLInfoData', 'record InfoCode : byte; InfoLen : Word; St'
13112: +'atementType : TALFBXStatementType; end
13113: FindClass('TOBJECT'), 'TALFBXLibrary
13114: //PALFBXStatusVector', '^PALFBXStatusVector // will not work
13115: TALFBXOnConnectionLost', 'Procedure ( Lib : TALFBXLibrary )
13116: //TALFBXOnGetDBExceptionClass', 'Procedure ( Number : Integer; out'
13117: +' Excep : EALFBXExceptionClass)
13118: SIRегистер_TALFBXLibrary(CL);
13119: 'cALFBXDateOffset', 'LongInt'( 15018 );
13120: 'cALFBXTimeCoeff', 'LongInt'( 864000000 );
13121: //Procedure ALFBXDecodeTimeStamp( v : PISCTimeStamp; out DateTime : Double);
13122: //Procedure ALFBXDecodeTimeStamp1( v : PISCTimeStamp; out TimeStamp : TTimeStamp);
13123: //Function ALFBXDecodeTimeStamp2( v : PISCTimeStamp ) : Double;
13124: Procedure ALFBXDecodeSQLDate( v : Integer; out Year : SmallInt; out Month, Day : Word )
13125: Procedure ALFBXDecodeSQLTime(v:Cardinal;out Hour,Minute,Second:Word; out Fractions: LongWord)
13126: //Procedure ALFBXEncodeTimeStamp( const DateTime : TDateTime; v : PISCTimeStamp );
13127: //Procedure ALFBXEncodeTimeStamp1( const Date : Integer; v : PISCTimeStamp );
13128: //Procedure ALFBXEncodeTimeStamp2( const Time : Cardinal; v : PISCTimeStamp );
13129: Function ALFBXEncodeSQLDate( Year : Integer; Month, Day : Integer ) : Integer
13130: Function ALFBXEncodeSQLTime( Hour, Minute, Second : Word; var Fractions : LongWord) : Cardinal
13131: TALFBXParamType', '( prNone, prByte, prShrt, prCard, prStrg, prLgno )
13132: TALFBXDPBInfo', 'record Name : AnsiString; ParamType : TALFBXParamType; end
13133: Function ALFBXSQLQuote( const name : AnsiString ) : AnsiString
13134: Function ALFBXSQLUnQuote( const name : AnsiString ) : AnsiString
13135: end;
13136:
13137: procedure SIRегистер_ALFBXClient(CL: TPSPPascalCompiler);
13138: begin
13139:   TALFBXClientSQLParam', 'record Value : AnsiString; IsNull : Boolean; end
13140:   TALFBXClientSQLParams', 'array of TALFBXClientSQLParam
13141:   TALFBXClientSelectDataSQL', 'record SQL : AnsiString; Params : T'
13142:   +'ALFBXClientSQLParams; RowTag : AnsiString; ViewTag : AnsiString; Skip : in'
13143:   +'teger; First : Integer; CacheThreshold : Integer; end
13144:   TALFBXClientSelectDataSQLs', 'array of TALFBXClientSelectDataSQL
13145:   TALFBXClientUpdateDataSQL', 'record SQL : AnsiString; Params: TALFBXClientSQLParams; end
13146:   TALFBXClientUpdateDataSQLs', 'array of TALFBXClientUpdateDataSQL
13147:   TALFBXClientMonitoringIOStats', 'record page_reads : int64; page'
13148:   +'_writes : int64; page_fetches : int64; page_marks : int64; end
13149:   SIRегистер_TALFBXClient(CL);
13150:   SIRегистер_TALFBXConnectionStatementPoolBinTreeNode(CL);
13151:   SIRегистер_TALFBXConnectionStatementPoolBinTree(CL);
13152:   SIRегистер_TALFBXConnectionWithStmtPoolContainer(CL);
13153:   SIRегистер_TALFBXConnectionWithoutStmtPoolContainer(CL);
13154:   SIRегистер_TALFBXReadTransactionPoolContainer(CL);
13155:   SIRегистер_TALFBXReadStatementPoolContainer(CL);
13156:   SIRегистер_TALFBXStringKeyPoolBinTreeNode(CL);
13157:   SIRегистер_TALFBXConnectionPoolClient(CL);
13158:   SIRегистер_TALFBXEventThread(CL);
13159:   Function AlMySqlClientSlashedStr( const Str : AnsiString ) : AnsiString
13160: end;
13161:
13162: procedure SIRегистер_ovcBidi(CL: TPSPPascalCompiler);
13163: begin
13164:   _OSVERSIONINFOA = record
13165:     dwOSVersionInfoSize: DWORD;
13166:     dwMajorVersion: DWORD;
13167:     dwMinorVersion: DWORD;
13168:     dwBuildNumber: DWORD;
13169:     dwPlatformId: DWORD;
13170:     szCSDVersion: array[0..127] of AnsiChar; { Maintenance AnsiString for PSS usage }
13171:   end;
13172:   TOSVersionInfoA', '_OSVERSIONINFOA
13173:   TOSVersionInfo', 'TOSVersionInfoA
13174:   'WS_EX_RIGHT', 'LongWord')($00001000);
13175:   'WS_EX_LEFT', 'LongWord')($00000000);
13176:   'WS_EX_RTLREADING', 'LongWord')($00002000);
13177:   'WS_EX_LTRREADING', 'LongWord')($00000000);
13178:   'WS_EX_LEFTSCROLLBAR', 'LongWord')($00004000);
13179:   'WS_EX_RIGHTSCROLLBAR', 'LongWord')($00000000);
13180:   Function SetProcessDefaultLayout( dwDefaultLayout : DWORD ) : BOOL
13181:   'LAYOUTRTL', 'LongWord')($00000001);
13182:   'LAYOUTBTT', 'LongWord')($00000002);
13183:   'LAYOUTVBH', 'LongWord')($00000004);

```

```

13184:   'LAYOUT_BITMAPORIENTATIONPRESERVED','LongWord')( $00000008);
13185:   'NOMIRRORBITMAP','LongWord')( DWORD ( $80000000 ));
13186:   Function SetLayout( dc : HDC; dwLayout : DWORD ) : DWORD
13187:   Function GetLayout( dc : hdc ) : DWORD
13188:   Function IsBidi : Boolean
13189:   Function GetCurrentHwProfile( var lpHwProfileInfo : THWProfileInfo ) : BOOL
13190:   Function GetVersionEx( var lpVersionInformation : TOSVersionInfo ) : BOOL
13191:   Function SetPriorityClass( hProcess : THandle; dwPriorityClass: DWORD ) : BOOL
13192:   Function GetPriorityClass( hProcess : THandle ) : DWORD
13193:   Function OpenClipboard( hWndNewOwner : HWND ) : BOOL
13194:   Function CloseClipboard : BOOL
13195:   Function GetClipboardSequenceNumber : DWORD
13196:   Function GetClipboardOwner : HWND
13197:   Function SetClipboardViewer( hWndNewViewer : HWND ) : HWND
13198:   Function GetClipboardViewer : HWND
13199:   Function ChangeClipboardChain( hWndRemove, hWndNewNext : HWND ) : BOOL
13200:   Function SetClipboardData( uFormat : UINT; hMem : THandle ) : THandle
13201:   Function GetClipboardData( uFormat : UINT ) : THandle
13202:   Function RegisterClipboardFormat( lpszFormat : PChar ) : UINT
13203:   Function CountClipboardFormats : Integer
13204:   Function EnumClipboardFormats( format : UINT ) : UINT
13205:   Function GetClipboardFormatName(format:UINT;lpszFormatName:PChar;cchMaxCount:Integer):Integer
13206:   Function EmptyClipboard : BOOL
13207:   Function IsClipboardFormatAvailable( format : UINT ) : BOOL
13208:     Function GetPriorityClipboardFormat( var paFormatPriorityList, cFormats : Integer ) : Integer
13209:   Function GetOpenClipboardWindow : HWND
13210:   Function EndDialog( hDlg : HWND; nResult : Integer ) : BOOL
13211:   Function GetDlgItem( hDlg : HWND; nIDDlgItem : Integer ) : HWND
13212:   Function SetDlgItemInt( hDlg : HWND; nIDDlgItem : Integer; uValue : UInt; bSigned: BOOL): BOOL
13213:   Function GetDlgItemInt(hDlg:HWND;nIDDlgItem:Integer;var lpTranslated:BOOL;bSigned: BOOL): UInt
13214:   Function SetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PChar ) : BOOL
13215:   Function CheckDlgButton( hDlg : HWND; nIDButton : Integer; uCheck : UInt ) : BOOL
13216:   Function CheckRadioButton( hDlg : HWND; nIDFirstButton, nIDLastButton,nIDCheckButton:Integer ) : BOOL
13217:   Function IsDlgButtonChecked( hDlg : HWND; nIDButton : Integer ) : UInt
13218:   Function SendDlgItemMessage(hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13219: end;
13220:
13221: procedure SIRegister_DXPUtils(CL: TPSPascalCompiler);
13222: begin
13223:   Function glExecuteAndWait( cmdLine:string;visibility:Word;timeout:Cardinal;killAppOnTimeOut:Bool ):Int;
13224:   Function GetTemporaryFilesPath : String
13225:   Function GetTemporaryFileName : String
13226:   Function FindfileInPaths( const fileName, paths : String ) : String
13227:   Function PathsToString( const paths : TStrings ) : String
13228:   Procedure StringToPaths( const pathsString : String; paths : TStrings )
13229: //Function MacroExpandPath( const aPath : String ) : String
13230: end;
13231:
13232: procedure SIRegister_ALMultiPartBaseParser(CL: TPSPascalCompiler);
13233: begin
13234:   SIRegister_TALMultiPartBaseContent(CL);
13235:   SIRegister_TALMultiPartBaseContents(CL);
13236:   SIRegister_TALMultiPartBaseStream(CL);
13237:   SIRegister_TALMultiPartBaseEncoder(CL);
13238:   SIRegister_TALMultiPartBaseDecoder(CL);
13239:   Function ALMultipartExtractBoundaryFromContentType( aContentType : AnsiString ) : AnsiString
13240:   Function ALMultipartExtractSubValueFromHeaderLine(aHeaderLine:AnsiString;aName:AnsiString):AnsiString;
13241:   Function ALMultipartSetSubValueInHeaderLine(aHeaderLine:AnsiString;aName,AValue:AnsiString):AnsiString;
13242: end;
13243:
13244: procedure SIRegister_SmallUtils(CL: TPSPascalCompiler);
13245: begin
13246:   TdriveSize', 'record FreeS : Int64; TotalS : Int64; end
13247:   TWinVerRec', 'record WinPlatform : Integer; WinMajorVersion : In-
13248:   +teger; WinMinorVersion : Integer; WinBuildNumber : Integer; WinCSDVersion: String; end
13249:   Function aAllocPadedMem( Size : Cardinal ) : TObject
13250:   Procedure aFreePadedMem( var P : TObject );
13251:   Procedure aFreePadedMem( var P : PChar );
13252:   Function aCheckPadedMem( P : Pointer ) : Byte
13253:   Function aGetPadMemSize( P : Pointer ) : Cardinal
13254:   Function aAllocMem( Size : Cardinal ) : Pointer
13255:   Function aStrLen( const Str : PChar ) : Cardinal
13256:   Function aStrLCopy( Dest : PChar; const Source : PChar; MaxLen : Cardinal ) : PChar
13257:   Function aStrECopy( Dest : PChar; const Source : PChar ) : PChar
13258:   Function aStrCopy( Dest : PChar; const Source : PChar ) : PChar
13259:   Function aStrEnd( const Str : PChar ) : PChar
13260:   Function aStrScan( const Str : PChar; aChr : Char ) : PChar
13261:   Function aStrMove( Dest : PChar; const Source : PChar; Count : Cardinal ) : PChar
13262:   Function aPCharLength( const Str : PChar ) : Cardinal
13263:   Function aPCharUpper( Str : PChar ) : PChar
13264:   Function aPCharLower( Str : PChar ) : PChar
13265:   Function aStrCat( Dest : PChar; const Source : PChar ) : PChar
13266:   Function aLastDelimiter( const Delimiters, S : String ) : Integer
13267:   Function aCopyTail( const S : String; Len : Integer ) : String
13268:   Function aInt2Thos( I : Int64 ) : String
13269:   Function aUpperCase( const S : String ) : String
13270:   Function aLowerCase( const S : string ) : String
13271:   Function aCompareText( const S1, S2 : string ) : Integer
13272:   Function aSameText( const S1, S2 : string ) : Boolean

```

```

13273: Function aInt2Str( Value : Int64 ) : String
13274: Function aStr2Int( const Value : String ) : Int64
13275: Function aStr2IntDef( const S : string; Default : Int64 ) : Int64
13276: Function aGetFileExt( const FileName : String ) : String
13277: Function aGetFilePath( const FileName : String ) : String
13278: Function aGetFileName( const FileName : String ) : String
13279: Function aChangeExt( const FileName, Extension : String ) : String
13280: Function aAdjustLineBreaks( const S : string ) : string
13281: Function aGetWindowStr( WinHandle : HWND ) : String
13282: Function aDiskSpace( Drive : String ) : TdriveSize
13283: Function aFileExists( FileName : String ) : Boolean
13284: Function aFileSize( FileName : String ) : Int64
13285: Function aDirectoryExists( const Name : string ) : Boolean
13286: Function aSysErrorMessage( ErrorCode : Integer ) : string
13287: Function aShortPathName( const LongName : string ) : string
13288: Function aGetWindowVer : TWinVerRec
13289: procedure InitDriveSpacePtr;
13290: end;
13291:
13292: procedure SIRegister_MakeApp(CL: TPSPascalCompiler);
13293: begin
13294:   aZero', 'LongInt'( 0 );
13295:   'makeappDEF', 'LongInt'( - 1 );
13296:   'CS_VREDRAW', 'LongInt'( DWORD( 1 ) );
13297:   'CS_HREDRAW', 'LongInt'( DWORD( 2 ) );
13298:   'CS_KEYCWTWINDOW', 'LongInt'( 4 );
13299:   'CS_DBLCLKS', 'LongInt'( 8 );
13300:   'CS_OWNDC', 'LongWord' )( $20 );
13301:   'CS_CLASSDC', 'LongWord' )( $40 );
13302:   'CS_PARENTDC', 'LongWord' )( $80 );
13303:   'CS_NOKEYCWT', 'LongWord' )( $100 );
13304:   'CS_NOCLOSE', 'LongWord' )( $200 );
13305:   'CS_SAVEBITS', 'LongWord' )( $800 );
13306:   'CS_BYTEALIGNCLIENT', 'LongWord' )( $1000 );
13307:   'CS_BYTEALIGNWINDOW', 'LongWord' )( $2000 );
13308:   'CS_GLOBALCLASS', 'LongWord' )( $4000 );
13309:   'CS_IME', 'LongWord' )( $10000 );
13310:   'CS_DROPSHADOW', 'LongWord' )( $20000 );
13311:   //TPanelFunc', 'TPanelFunc // will not work
13312:   TPanelStyle', '(psEdge, psTabEdge, psBorder, psTabBorder, psTab, psNone )
13313:   TFontLook', '( flBold, flItalic, flUnderLine, flStrikeOut )
13314:   TFontLooks', 'set of TFontLook
13315:   TMessagefunc', 'function(hWnd,iMsg,wParam,lParam:Integer):Integer
13316:   Function SetWinClass( const ClassName:String; pMessFunc: Tmessagefunc; wcStyle : Integer): Word
13317:   Function SetWinClassO( const ClassName : String; pMessFunc : TObject; wcStyle : Integer): Word
13318:   Function SetWinClass( const ClassName : String; pMessFunc : TObject; wcStyle : Integer ) : Word
13319:   Function MakeForm(Left,Top,Width,Height:Integer;const Caption:String;WinStyle:Integer):Integer
13320:   Procedure RunMsgLoop( Show : Boolean )
13321:   Function MakeFont(Height,Width:Integer; const FontName:String; Look:TFontLooks; Roman:Boolean): Integer
13322:   Function MakeButton(Left,Top,Width,Height:Integer;pCaption:PChar;hParent,
ID_Number:Cardinal;hFont:Int):Int;
13323:   Function MakeListBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Integer
13324:   Function MakeComboBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Int
13325:   Function MakePanel(Left,Top,Width,Height,
hParent:Int;WndFunc:TPanelFunc;ID_Number:Card;Style:TPanelStyle):Int;
13326:   Function MakeSubMenu(const ItemList : String; ID1, ID2 : Cardinal; hMenu : Integer) : Integer
13327:   Function id4menu( a, b : Byte; c : Byte; d : Byte ) : Cardinal
13328:   Procedure DoInitMakeApp //set first to init formclasscontrol!
13329: end;
13330:
13331: procedure SIRegister_ScreenSaver(CL: TPSPascalCompiler);
13332: begin
13333:   TScreenSaverOption', '( ssoAutoAdjustFormProperties, ssoAutoHook'
13334:     + 'KeyboardEvents, ssoAutoHookMouseEvents, ssoEnhancedMouseMoveDetection )'
13335:   TScreenSaverOptions', 'set of TScreenSaverOption
13336:   'cDefaultScreenSaverOptions', 'LongInt' ).Value.ts32:=ord(ssoAutoAdjustFormProperties) or
ord(ssoAutoHookKeyboardEvents) or ord(ssoEnhancedMouseMoveDetection)
13337:   TScreenSaverPreviewEvent', 'Procedure ( Sender : TObject; previewHwnd: HWND )
13338:   SIRegister_TScreenSaver(CL);
13339:   //Procedure Register
13340:   Procedure SetScreenSaverPassword
13341: end;
13342:
13343: procedure SIRegister_XCollection(CL: TPSPascalCompiler);
13344: begin
13345:   FindClass('TOBJECT'), 'TXCollection
13346:   SIRegister_EFilerException(CL);
13347:   SIRegister_TXCollectionItem(CL);
13348:   //TXCollectionItemClass', 'class of TXCollectionItem
13349:   SIRegister_TXCollection(CL);
13350:   Procedure RegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent )
13351:   Procedure DeRegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent )
13352:   Procedure RegisterXCollectionItemClass( aClass : TXCollectionItemClass )
13353:   Procedure UnregisterXCollectionItemClass( aClass : TXCollectionItemClass )
13354:   Function FindXCollectionItemClass( const className : String ) : TXCollectionItemClass
13355:   Function GetXCollectionItemClassesList( baseClass : TXCollectionItemClass ) : TList
13356: end;
13357:
13358: procedure SIRegister_XOpenGL(CL: TPSPascalCompiler);

```

```

13359: begin
13360:   TMapTexCoordMode', '(mtcmUndefined, mtcmNull, mtcmMain, mtcmDual, mtcmSecond, mtcmArbitrary);
13361:   Procedure xglMapTexCoordToNull
13362:   Procedure xglMapTexCoordToMain
13363:   Procedure xglMapTexCoordToSecond
13364:   Procedure xglMapTexCoordToDual
13365:   Procedure xglMapTexCoordToArbitrary( const units : array of Cardinal );
13366:   Procedure xglMapTexCoordToArbitrary1( const bitWiseUnits : Cardinal );
13367:   Procedure xglMapTexCoordToArbitraryAdd( const bitWiseUnits : Cardinal )
13368:   Procedure xglBeginUpdate
13369:   Procedure xglEndUpdate
13370:   Procedure xglPushState
13371:   Procedure xglPopState
13372:   Procedure xglForbidSecondTextureUnit
13373:   Procedure xglAllowSecondTextureUnit
13374:   Function xglGetBitWiseMapping : Cardinal
13375: end;
13376:
13377: procedure SIRegister_VectorLists(CL: TPSPascalCompiler);
13378: begin
13379:   TBaseListOption', '( bloExternalMemory, bloSetCountResetsMemory )
13380:   TBaseListOptions', 'set of TBaseListOption
13381:   SIRegister_TBaseList(CL);
13382:   SIRegister_TBaseVectorList(CL);
13383:   SIRegister_TAffineVectorList(CL);
13384:   SIRegister_TVectorList(CL);
13385:   SIRegister_TTexPointList(CL);
13386:   SIRegister_TXIntegerList(CL);
13387: //PSingleArrayList', '^TSingleArrayList // will not work
13388:   SIRegister_TSingleList(CL);
13389:   SIRegister_TByteList(CL);
13390:   SIRegister_TQuaternionList(CL);
13391:   Procedure QuickSortLists( startIndex, endIndex : Integer; refList : TSingleList; objList : TList );
13392:   Procedure QuickSortLists1( startIndex, endIndex : Integer; refList : TSingleList; objList : TBaseList );
13393:   Procedure FastQuickSortLists(startIndex,
endIndex:Integer;refList:TSingleList;objList:TPersistentObjectList);
13394: end;
13395:
13396: procedure SIRegister_MeshUtils(CL: TPSPascalCompiler);
13397: begin
13398:   Procedure ConvertStripToList( const strip : TAffineVectorList; list : TAffineVectorList );
13399:   Procedure ConvertStripToList1( const strip : TIntegerList; list : TIntegerList );
13400:   Procedure ConvertStripToList2(const strip:TAffineVectorList;const
indices:TIntegerList;list:TAffineVectorList);
13401:   Procedure ConvertIndexedListToList(const data:TAffineVectlist;const
indices:TIntegerList;list:TAffineVectorList);
13402:   Function BuildVectorCountOptimizedIndices(const vertices:TAffineVectorList; const
normals:TAffineVectorList; const texCoords : TAffineVectorList ) : TIntegerList
13403:   Procedure RemapReferences( reference : TAffineVectorList; const indices : TIntegerList );
13404:   Procedure RemapReferences1( reference : TIntegerList; const indices : TIntegerList );
13405:   Procedure RemapAndCleanupReferences( reference : TAffineVectorList; indices : TIntegerList );
13406:   Function RemapIndicesToIndicesMap( remapIndices : TIntegerList ) : TIntegerList
13407:   Procedure RemapTrianglesIndices( indices, indicesMap : TIntegerList );
13408:   Procedure RemapIndices( indices, indicesMap : TIntegerList );
13409:   Procedure UnifyTrianglesWinding( indices : TIntegerList );
13410:   Procedure InvertTrianglesWinding( indices : TIntegerList );
13411:   Function BuildNormals( reference : TAffineVectorList; indices : TIntegerList ) : TAffineVectorList
13412:   Function BuildNonOrientedEdgesList(triangleIndices:TIntegerList; triangleEdges : TIntegerList;
edgesTriangles : TIntegerList) : TIntegerList
13413:   Procedure WeldVertices( vertices : TAffineVectorList; indicesMap : TIntegerList; weldRadius : Single )
13414:   Function StripifyMesh(indices:TIntegerList;maxVertexIndex:Integer;agglomerateLoneTriangles:Boolean):TPersistentObjectList;
13415:   Procedure IncreaseCoherency( indices : TIntegerList; cacheSize : Integer )
13416:   Procedure SubdivideTriangles( smoothFactor : Single; vertices : TAffineVectorList; triangleIndices : TIntegerList;
normals : TAffineVectorList; onSubdivideEdge : TSubdivideEdgeEvent )
13417: end;
13418:
13419: procedure SIRegister_JclSysUtils(CL: TPSPascalCompiler);
13420: begin
13421:   Procedure GetAndFillMem( var P : TObject; const Size : Integer; const Value : Byte )
13422:   Procedure FreeMemAndNil( var P : TObject )
13423:   Function PCharOrNil( const S : string ) : PChar
13424:   SIRegister_TJclReferenceMemoryStream(CL);
13425:   FindClass('TOBJECT'), 'EJclVMTError
13426:   {Function GetVirtualMethodCount( AClass : TClass ) : Integer
13427:   Function GetVirtualMethod( AClass : TClass; const Index : Integer ) : Pointer
13428:   Procedure SetVirtualMethod( AClass : TClass; const Index : Integer; const Method:Pointer )
13429:     PDYNAMICINDEXLIST', '^TDYNAMICINDEXLIST // will not work
13430:     PDYNAMICADDRESSLIST', '^TDYNAMICADDRESSLIST // will not work
13431:   Function GetDynamicMethodCount( AClass : TClass ) : Integer
13432:   Function GetDynamicIndexList( AClass : TClass ) : PDYNAMICINDEXLIST
13433:   Function GetDynamicAddressList( AClass : TClass ) : PDYNAMICADDRESSLIST
13434:   Function HasDynamicMethod( AClass : TClass; Index : Integer ) : Boolean
13435:   Function GetDynamicMethod( AClass : TClass; Index : Integer ) : Pointer
13436:   Function GetInitTable( AClass : TClass ) : PTyepInfo
13437:     PFIELDENTRY', '^TFieldEntry // will not work}
13438:   TFieldEntry', 'record Offset : Integer; IDX : Word; Name : ShortString; end
13439:   Function JIsClass( Address : Pointer ) : Boolean
13440:   Function JIsObject( Address : Pointer ) : Boolean

```

```

13441: Function GetImplementorOfInterface( const I : IIInterface ) : TObject
13442:   TDigitCount', 'Integer
13443:   SIRegister_TJclNumericFormat(CL);
13444: Function JIntToStrZeroPad( Value, Count : Integer ) : AnsiString
13445:   TTextHandler', 'Procedure ( const Text : string )
13446: // 'ABORT_EXIT_CODE','LongInt'( ERROR_CANCELLED 1223 );
13447: Function JExecute(const
13448:   CommandLine:string;OutputLineCallback:TTextHandler;RawOutput:Bool;AbortPtr:PBool):Cardinal;
13449: Function ReadKey : Char //to and from the DOS console !
13450:   TModuleHandle', 'HINST
13451: //TModuleHandle', 'Pointer
13452: 'INVALID_MODULEHANDLE_VALUE','LongInt'( TModuleHandle ( 0 ) );
13453: Function LoadModule( var Module : TModuleHandle; FileName : string ) : Boolean
13454: Function LoadModuleEx( var Module : TModuleHandle; FileName : string; Flags : Cardinal ) : Boolean
13455: Procedure UnloadModule( var Module : TModuleHandle )
13456: Function GetModuleSymbol( Module : TModuleHandle; SymbolName : string ) : Pointer
13457: Function GetModuleSymbolEx( Module : TModuleHandle; SymbolName : string; var Accu : Boolean ) : Pointer
13458: Function ReadModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size: Cardinal):Boolean;
13459: Function WriteModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size:Cardinal):Boolean;
13460: FindClass('TOBJECT'),'EJclConversionError
13461: Function JStrToBoolean( const S : string ) : Boolean
13462: Function JBooleanToStr( B : Boolean ) : string
13463: Function JIntToBool( I : Integer ) : Boolean
13464: Function JBoolToInt( B : Boolean ) : Integer
13465: 'ListSeparator','String ';
13466: 'ListSeparator1','String ';
13467: Procedure ListAddItems( var List : string; const Separator, Items : string )
13468: Procedure ListIncludeItems( var List : string; const Separator, Items : string )
13469: Procedure ListRemoveItems( var List : string; const Separator, Items : string )
13470: Procedure ListDeleteItem( var List : string; const Separator : string; const Index : Integer )
13471: Function ListItemCount( const List, Separator : string ) : Integer
13472: Function ListGetItem( const List, Separator : string; const Index : Integer ) : string
13473: Procedure ListSetItem(var List:string;const Separator:string;const Index:Integer;const Value:string)
13474: Function ListItemIndex( const List, Separator, Item : string ) : Integer
13475: Function SystemTOBJECTInstance : LongWord
13476: Function IsCompiledWithPackages : Boolean
13477: Function JJclGUIDToString( const GUID : TGUID ) : string
13478: Function JJclStringToGUID( const S : string ) : TGUID
13479: SIRegister_TJclIntfCriticalSection(CL);
13480: SIRegister_TJclSimpleLog(CL);
13481: Procedure InitSimpleLog( const ALogFileFileName : string )
13482: end;
13483:
13484: procedure SIRegister_JclBorlandTools(CL: TPSPascalCompiler);
13485: begin
13486:   FindClass('TOBJECT'),'EJclBorRADException
13487:   TJclBorRADToolKind', '( brDelphi, brCppBuilder, brBorlandDevStudio )
13488:   TJclBorRADToolEdition', '( deOPEN, dePRO, deSVR )
13489:   TJclBorRADToolEdition', '( deSTD, dePRO, deCSS, deARC )
13490:   TJclBorRADToolPath', 'string
13491: 'SupportedDelphiVersions','LongInt'( 5 or 6 or 7 or 8 or 9 or 10 or 11 );
13492: 'SupportedBCBVersions','LongInt'( 5 or 6 or 10 or 11 );
13493: 'SupportedBDSVersions','LongInt'( 1 or 2 or 3 or 4 or 5 );
13494: BorRADToolRepositoryPagesSection', 'String 'Repository Pages
13495: BorRADToolRepositoryDialogsPage', 'String 'Dialogs
13496: BorRADToolRepositoryFormsPage', 'String 'Forms
13497: BorRADToolRepositoryProjectsPage', 'String 'Projects
13498: BorRADToolRepositoryDataModulesPage', 'String 'Data Modules
13499: BorRADToolRepositoryObjectType', 'String 'Type
13500: BorRADToolRepositoryFormTemplate', 'String 'FormTemplate
13501: BorRADToolRepositoryProjectTemplate', 'String 'ProjectTemplate
13502: BorRADToolRepositoryObjectName', 'String 'Name
13503: BorRADToolRepositoryObjectPage', 'String 'Page
13504: BorRADToolRepositoryObjectIcon', 'String 'Icon
13505: BorRADToolRepositoryObjectDescr', 'String 'Description
13506: BorRADToolRepositoryObjectAuthor', 'String 'Author
13507: BorRADToolRepositoryObjectAncestor', 'String 'Ancestor
13508: BorRADToolRepositoryObjectDesigner', 'String 'Designer
13509: BorRADToolRepositoryDesignerDfm', 'String 'dfm
13510: BorRADToolRepositoryDesignerXfm', 'String 'xmf
13511: BorRADToolRepositoryObjectNewForm', 'String 'DefaultNewForm
13512: BorRADToolRepositoryObject MainForm', 'String 'DefaultMainForm
13513: SourceExtensionDelphiPackage', 'String '.dpk
13514: SourceExtensionBCBPackage', 'String '.bpk
13515: SourceExtensionDelphiProject', 'String '.dpr
13516: SourceExtensionBCBProject', 'String '.bpr
13517: SourceExtensionBDSProject', 'String '.bdsproj
13518: SourceExtensionDProject', 'String '.dproj
13519: BinaryExtensionPackage', 'String '.bpl
13520: BinaryExtensionLibrary', 'String '.dll
13521: BinaryExtensionExecutable', 'String '.exe
13522: CompilerExtensionDCP', 'String '.dep
13523: CompilerExtensionBPI', 'String '.bpi
13524: CompilerExtensionLIB', 'String '.lib
13525: CompilerExtensionTDS', 'String '.tds
13526: CompilerExtensionMAP', 'String '.map
13527: CompilerExtensionDRC', 'String '.drc
13528: CompilerExtensionDEF', 'String '.def

```

```

13529: SourceExtensionCPP', 'String '.cpp
13530: SourceExtensionH', 'String '.h
13531: SourceExtensionPAS', 'String '.pas
13532: SourceExtensionDFM', 'String '.dfm
13533: SourceExtensionXFM', 'String '.xfm
13534: SourceDescriptionPAS', 'String 'Pascal source file
13535: SourceDescriptionCPP', 'String 'C++ source file
13536: DesignerVCL', 'String 'VCL
13537: DesignerCLX', 'String 'CLX
13538: ProjectTypePackage', 'String 'package
13539: ProjectTypeLibrary', 'String 'library
13540: ProjectTypeProgram', 'String 'program
13541: Personality32Bit', 'String '32 bit
13542: Personality64bit', 'String '64 bit
13543: PersonalityDelphi', 'String 'Delphi
13544: PersonalityDelphiDotNet', 'String 'Delphi.net
13545: PersonalityBCB', 'String 'C++Builder
13546: PersonalityCSB', 'String 'C#Builder
13547: PersonalityVB', 'String 'Visual Basic
13548: PersonalityDesign', 'String 'Design
13549: PersonalityUnknown', 'String 'Unknown personality
13550: PersonalityBDS', 'String 'Borland Developer Studio
13551: DOFDirectoriesSection', 'string 'Directories
13552: DOFUUnitOutputDirKey', 'String 'UnitOutputDir
13553: DOFSearchPathName', 'String 'SearchPath
13554: DOFConditionals', 'String 'Conditionals
13555: DOFLinkerSection', 'String 'Linker
13556: DOFPackagesKey', 'String 'Packages
13557: DOFCompilerSection', 'String 'Compiler
13558: DOFPackageNoLinkKey', 'String 'PackageNoLink
13559: DOFAdditionalSection', 'String 'Additional
13560: DOFOptionsKey', 'String 'Options
13561: TJclBorPersonality', '( bpDelphi32, bpDelphi64, bpBCBuilder32, b'
13562: +'pBCBuilder64, bpDelphiNet32, bpDelphiNet64, bpCSBuilder32, bpCSBuilder64, '
13563: +'bpVisualBasic32, bpVisualBasic64, bpDesign, bpUnknown )
13564: TJclBorPersonalities', 'set of TJclBorPersonality
13565: TJclBorDesigner', '( bdVCL, bdCLX )
13566: TJclBorDesigners', 'set of TJclBorDesigner
13567: TJclBorPlatform', '( bp32bit, bp64bit )
13568: FindClass('TOBJECT'), TJclBorRADToolInstallation
13569: SIRegister_TJclBorLandOpenHelp(CL);
13570: SIRegister_TJclBorHelp2Object(CL);
13571: TJclHelp2Object', '( hoRegisterSession, hoRegister, hoPlugin )
13572: TJclHelp2Objects', 'set of TJclHelp2Object
13573: SIRegister_TJclHelp2Manager(CL);
13574: SIRegister_TJclBorRADToolIDETool(CL);
13575: SIRegister_TJclBorRADToolIDEPackages(CL);
13576: SIRegister_IJclCommandLineTool(CL);
13577: FindClass('TOBJECT'), EJclCommandLineToolError
13578: SIRegister_TJclCommandLineTool(CL);
13579: SIRegister_TJclBorLandCommandLineTool(CL);
13580: SIRegister_TJclBCC32(CL);
13581: SIRegister_TJclDCC32(CL);
13582: TJclDCC', 'TJclDCC32
13583: SIRegister_TJclBpr2Mak(CL);
13584: SIRegister_TJclBorLandMake(CL);
13585: SIRegister_TJclBorRADToolPalette(CL);
13586: SIRegister_TJclBorRADToolRepository(CL);
13587: TCommandLineTool', '( clAsm, clBcc32, clDcc32, clDccIL, clMake, clProj2Mak )
13588: TCommandLineTools', 'set of TCommandLineTool
13589: //TJclBorRADToolInstallationClass', 'class of TJclBorRADToolInstallation
13590: SIRegister_TJclBorRADToolInstallation(CL);
13591: SIRegister_TJclBCBInstallation(CL);
13592: SIRegister_TJclDelphiInstallation(CL);
13593: SIRegister_TJclDCCIL(CL);
13594: SIRegister_TJclBDSInstallation(CL);
13595: TTraverseMethod', 'Function ( Installation : TJclBorRADToolInstallation ) : Boolean
13596: SIRegister_TJclBorRADToolInstallations(CL);
13597: Function BPLFileName( const BPLPath, PackageFileName : string ) : string
13598: Function BinaryFileName( const OutputPath, ProjectFileName : string ) : string
13599: Function IsDelphiPackage( const FileName : string ) : Boolean
13600: Function IsDelphiProject( const FileName : string ) : Boolean
13601: Function IsBCBPackage( const FileName : string ) : Boolean
13602: Function IsBCBProject( const FileName : string ) : Boolean
13603: Procedure GetDPRFileInfo( const DPRFileName:string;out BinaryExtensio:string;const LibSuffix:PString );
13604: Procedure GetBPRFileInfo( const BPRFileName:string;out BinaryFileName:string;const Descript:PString );
13605: Procedure GetDPKFileInfo( const DPKFileName:string;out RunOnly:Bool;const LibSuffix:PString;const
Descript:PString;
13606: Procedure GetBPKFileInfo( const BPKFileName:string;out RunOnly:Bool;const BinaryFName:PString;const
Descript:PString
13607: function SamePath(const Path1, Path2: string): Boolean;
13608: end;
13609:
13610: procedure SIRegister_JclFileUtils_max(CL: TPSpascalCompiler);
13611: begin
13612:   'ERROR_NO_MORE_FILES', 'LongInt'( 18 );
13613:  //Function stat64( FileName: PChar;var StatBuffer : TStatBuf64 ) : Integer
13614:  //Function fstat64( FileDes: Integer;var StatBuffer : TStatBuf64 ) : Integer
13615:  //Function lstat64( FileName: PChar;var StatBuffer : TStatBuf64 ) : Integer

```

```

13616: 'LPathSeparator','String '/
13617: 'LDirDelimiter','String '/
13618: 'LDirSeparator','String ':'
13619: 'JXPathDevicePrefix','String '\\.\\
13620: 'JXPathSeparator','String '\
13621: 'JXDirDelimiter','String '\
13622: 'JXDirSeparator','String ';
13623: 'JXPathUncPrefix','String '\\\
13624: 'faNormalFile','LongWord')($00000080);
13625: //faUnixSpecific,'faSymLink);
13626: JXTCompactPath', '( cpCenter, cpEnd )
13627:     _WIN32_FILE_ATTRIBUTE_DATA', 'record dwFileAttributes : DWORD; f'
13628:     +'tCreationTime: TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime : '
13629:     +'TFileTime; nFileSizeHigh : DWORD; nFileSizeLow : DWORD; end
13630: TWin32FileAttributeData', '_WIN32_FILE_ATTRIBUTE_DATA'
13631: WIN32_FILE_ATTRIBUTE_DATA', '_WIN32_FILE_ATTRIBUTE_DATA
13632:
13633: Function jxPathAddSeparator( const Path : string ) : string
13634: Function jxPathAddExtension( const Path, Extension : string ) : string
13635: Function jxPathAppend( const Path, Append : string ) : string
13636: Function jxPathBuildRoot( const Drive : Byte) : string
13637: Function jxPathCanonicalize( const Path : string ) : string
13638: Function jxPathCommonPrefix( const Path1, Path2 : string ) : Integer
13639: Function jxPathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string
13640: Procedure jxPathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
13641: Function jxPathExtractFileDirFixed( const S : string ) : string
13642: Function jxPathExtractFileNameNoExt( const Path : string ) : string
13643: Function jxPathExtractPathDepth( const Path : string; Depth : Integer ) : string
13644: Function jxPathGetDepth( const Path : string ) : Integer
13645: Function jxPathGetLongName( const Path : string ) : string
13646: Function jxPathGetShortName( const Path : string ) : string
13647: Function jxPathGetLongName( const Path : string ) : string
13648: Function jxPathGetShortName( const Path : string ) : string
13649: Function jxPathGetRelativePath( Origin, Destination : string ) : string
13650: Function jxPathGetTempPath : string
13651: Function jxPathIsAbsolute( const Path : string ) : Boolean
13652: Function jxPathIsChild( const Path, Base : string ) : Boolean
13653: Function jxPathIsDiskDevice( const Path : string ) : Boolean
13654: Function jxPathIsUNC( const Path : string ) : Boolean
13655: Function jxPathRemoveSeparator( const Path : string ) : string
13656: Function jxPathRemoveExtension( const Path : string ) : string
13657: Function jxPathGetPhysicalPath( const LocalizedPath : string ) : string
13658: Function jxPathGetLocalizedPath( const PhysicalPath : string ) : string
13659: JxTFileListOption', '( flFullNames, flRecursive, flMaskedSubfolders )
13660: JxTFileListOptions', 'set of TFileListOption
13661: JxTJclAttributeMatch', '( amAny, amExact, amSubSetOf, amSuperSetOf, amCustom )
13662: TFileHandler', 'Procedure ( const FileName : string )
13663: TFileHandlerEx', 'Procedure ( const Directory : string; const FileInfo : TSearchRec)
13664: Function BuildFileList( const Path : string; const Attr : Integer; const List : TStrings ) : Boolean
13665: //Function AdvBuildFileList( const Path : string; const Attr : Integer; const Files : TStrings; const
13666: AttributeMatch:TJclAttributeMatch;const Optis:TFileListOptions;const SubfoldersMask:string;const
13667: FileMatchFunc:TFileMatchFunc):Bool;
13668: Function jxVerifyFileAttributeMask( var RejectedAttributes, RequiredAttributes : Int ) : Boolean
13669: Function jxIsFileAttributeMatch(FileAttributes,RejectedAttributes,RequiredAttributes:Int):Boolean;
13670: Function jxFileAttributesStr( const FileInfo : TSearchRec ) : string
13671: Function jxIsFileNameMatch(FileName:string;const Mask:string;const CaseSensitive:Boolean):Boolean;
13672: Procedure jxEnumFiles(const Path:string; HandleFile:TFileHandlerEx;
13673: RejectedAttributes:Integer;RequiredAttributes : Integer; Abort : TObject)
13674: Procedure jxEnumDirectories(const Root:string;const HandleDirectory:TFileHandler;const
13675: IncludeHiddenDirects:Boolean;const SubDirectoriesMask:string;Abort:TObject;ResolveSymLinks:Bool)
13676: Procedure jxCreatEmptyFile( const FileName : string )
13677: Function jxCloseVolume( var Volume : THandle ) : Boolean
13678: Function jxDeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean ) : Boolean
13679: Function jxCopyDirectory( ExistingDirectoryName, NewDirectoryName : string ) : Boolean
13680: Function jxMoveDirectory( ExistingDirectoryName, NewDirectoryName : string ) : Boolean
13681: Function jxDelTree( const Path : string ) : Boolean
13682: //Function DelTreeEx(const Path:string;AbortOnFailure:Boolean; Progress:TDelTreeProgress):Boolean
13683: Function jxDiskInDrive( Drive : Char ) : Boolean
13684: Function jxDirectoryExists( const Name : string; ResolveSymLinks : Boolean ) : Boolean
13685: Function jxFileCreateTemp( var Prefix : string ) : THandle
13686: Function jxFilBackup( const FileName : string; Move : Boolean ) : Boolean
13687: Function jxFilCopy( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean ) : Boolean
13688: Function jxFilDelete( const FileName : string; MoveToRecycleBin : Boolean ) : Boolean
13689: Function jxFilExists( const FileName : string ) : Boolean
13690: Function jxFilMove( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean ) : Boolean
13691: Function jxFilRestore( const FileName : string ) : Boolean
13692: Function jxFilGetBackupFileName( const FileName : string ) : string
13693: Function jxFilIsBackupFileName( const FileName : string ) : Boolean
13694: Function jxFilGetDisplayName( const FileName : string ) : string
13695: Function jxFilGetGroupName( const FileName : string; ResolveSymLinks : Boolean ) : string
13696: Function jxFilGetOwnerName( const FileName : string; ResolveSymLinks : Boolean ) : string
13697: Function jxFilGetSize( const FileName : string ) : Int64
13698: Function jxFilGetType( const Prefix : string ) : string
13699: Function jxFilGetTempName( const Prefix : string ) : string
13700: Function jxFilFindUnusedFileName(FileName:string; const FileExt : string; NumberPrefix : string ) : string
13701: Function jxFilForceDirectories( Name : string ) : Boolean
13702: Function jxFilGetDirectorySize( const Path : string ) : Int64
13703: Function jxFilGetDriveTypeStr( const Drive : Char ) : string
13704: Function jxFilGetFileAgeCoherence( const FileName : string ) : Boolean

```

```

13701: Procedure jxGetFileAttributeList( const Items : TStrings; const Attr : Integer)
13702: Procedure jxGetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
13703: Function jxGetFileInfo( const FileName : string; out FileInfo : TSearchRec ) : Boolean;
13704: Function jxGetInformation( const FileName : string ) : TSearchRec;
13705: //Function GetFileStatus(const FileName:string;out StatBuf:TStatBuf64;const
13706:   ResolveSymLinks:Boolean):Integer
13706: Function jxGetFileLastWrite( const FName : string ) : TFileTime;
13707: Function jxGetFileLastWrite1( const FName : string; out LocalTime : TDateTime ) : Boolean;
13708: Function jxGetFileLastAccess( const FName : string ) : TFileTime;
13709: Function jxGetFileLastAccess1( const FName : string; out LocalTime : TDateTime ) : Boolean;
13710: Function jxGetFileCreation( const FName : string ) : TFileTime;
13711: Function jxGetFileCreation1( const FName : string; out LocalTime : TDateTime ) : Boolean;
13712: Function jxGetFileLastWrite( const FName : string; out TimeStamp:Integer;ResolveSymLinks : Bool):Bool;
13713: Function jxGetFileLastWrite1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool): Bool;
13714: Function jxGetFileLastWrite2( const FName : string; ResolveSymLinks : Boolean ) : Integer;
13715: Function jxGetFileLastAccess(const FName:string; out TimeStamp:Integer;ResolveSymLinks: Bool): Bool;
13716: Function jxGetFileLastAccess1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool): Bool;
13717: Function jxGetFileLastAccess2(const FName:string; ResolveSymLinks:Boolean): Integer;
13718: Function jxGetFileLastAttrChange(const FName:string;out TimeStamp:Integer;ResolveSymLinks:Bool): Bool;
13719: Function jxGetFileLastAttrChang1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool): Bool;
13720: Function jxGetFileLastAttrChange2( const FName : string; ResolveSymLinks: Boolean ) : Integer;
13721: Function jxGetModulePath( const Module : HMODULE ) : string;
13722: Function jxGetSizeOfFile( const FileName : string ) : Int64;
13723: Function jxGetSizeOfFile1( const FileInfo : TSearchRec ) : Int64;
13724: Function jxGetSizeOfFile2( Handle : THandle ) : Int64;
13725: Function jxGetStandardFileInfo( const FileName : string ) : TWin32FileAttributeData;
13726: Function jxIsDirectory( const FileName : string; ResolveSymLinks : Boolean ) : Boolean;
13727: Function jxIsRootDirectory( const CanonicalFileName : string ) : Boolean;
13728: Function jxLockVolume( const Volume : string; var Handle : THandle ) : Boolean;
13729: Function jxOpenVolume( const Drive : Char ) : THandle;
13730: Function jxSetDirLastWrite( const DirName : string; const DateTime : TDateTime ) : Boolean;
13731: Function jxSetDirLastAccess( const DirName : string; const DateTime : TDateTime ) : Boolean;
13732: Function jxSetDirCreation( const DirName : string; const DateTime : TDateTime ) : Boolean;
13733: Function jxSetFileLastWrite( const FileName : string; const DateTime : TDateTime ) : Boolean;
13734: Function jxSetFileLastAccess( const FileName : string; const DateTime : TDateTime ) : Boolean;
13735: Function jxSetFileCreation( const FileName : string; const DateTime : TDateTime ) : Boolean;
13736: Procedure jxShredfile( const FileName : string; Times : Integer );
13737: Function jxUnlockVolume( var Handle : THandle ) : Boolean;
13738: Function jxCreatSymbolicLink( const Name, Target : string ) : Boolean;
13739: Function jxSymbolicLinkTarget( const Name : string ) : string;
13740: TAttributeInterest', '( aiIgnored, aiRejected, aiRequired )';
13741: SIRegister_TJclCustomFileAttrMask(CL);
13742: SIRegister_TJclFileAttributeMask(CL);
13743: TFileSearchOption', '( fsIncludeSubDirectories, fsIncludeHidden$,
13744: +'ubDirectories, fsLastChangeAfter, fsLastChangeBefore, fsMaxSize, fsMinSize)';
13745: TFileSearchOptions', 'set of TFileSearchOption';
13746: TFileSearchTaskID', 'Integer';
13747: TFileSearchTerminationEvent', 'Procedure ( const ID : TFileSearc';
13748: +'hTaskID; const Aborted : Boolean)';
13749: TFileEnumeratorSyncMode', '( smPerFile, smPerDirectory )';
13750: SIRegister_IJclFileEnumerator(CL);
13751: SIRegister_TJclFileEnumerator(CL);
13752: Function JxFileSearch : IJclFileEnumerator;
13753: JxTFileFlag', '( ffDebug, ffInfoInferred, ffPatched, ffPreRelease, ffPrivateBuild, ffSpecialBuild )';
13754: JxTFileFlags', 'set of TFileFlag';
13755: FindClass('TOBJECT'), 'EJclFileVersionInfoError';
13756: SIRegister_TJclFileVersionInfo(CL);
13757: Function jxOSIdentToString( const OSIdent : DWORD ) : string;
13758: Function jxOSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD ) : string;
13759: Function jxVersionResourceAvailable( const FileName : string ) : Boolean;
13760: TFileVersionFormat', '( vfMajorMinor, vfFull )';
13761: Function jxFormatVersionString( const HiV, LoV : Word ) : string;
13762: Function jxFormatVersionString1( const Major, Minor, Build, Revision : Word ) : string;
13763: //Function FormatVersionString2( const FixedInfo : TVSFixedFileInfo;VersionFormat:TFFileVersionFormat):str;
13764: //Procedure VersionExtractFileInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);
13765: //Procedure VersionExtractProductInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,
13766: Revision:Word);
13766: //Function VersionFixedFileInfo( const FileName : string; var FixedInfo : TVSFixedFileInfo ) : Boolean;
13767: Function jxVersionFixedFileInfoString( const FileName : string; VersionFormat : TFileVersionFormat; const
13768: NotAvailableText : string ) : string;
13768: SIRegister_TJclTempFileStream(CL);
13769: FindClass('TOBJECT'), 'TJclCustomFileMapping';
13770: SIRegister_TJclFileMappingView(CL);
13771: TJclFileMappingRoundOffset', '( rvDown, rvUp )';
13772: SIRegister_TJclCustomFileMapping(CL);
13773: SIRegister_TJclFileMapping(CL);
13774: SIRegister_TJclSwapFileMapping(CL);
13775: SIRegister_TJclFileMappingStream(CL);
13776: TJclMappedTextReaderIndex', '( tiNoIndex, tiFull )';
13777: //PPCharArray', '^TPCharArray // will not work
13778: SIRegister_TJclMappedTextReader(CL);
13779: SIRegister_TJclFileMaskComparator(CL);
13780: FindClass('TOBJECT'), 'EJclPathError';
13781: FindClass('TOBJECT'), 'EJclFileUtilsError';
13782: FindClass('TOBJECT'), 'EJclTempFileStreamError';
13783: FindClass('TOBJECT'), 'EJclTempFileStreamError';
13784: FindClass('TOBJECT'), 'EJclFileMappingError';
13785: FindClass('TOBJECT'), 'EJclFileMappingViewError';
13786: Function jxPathGetLongName2( const Path : string ) : string;

```

```

13787: Function jxWin32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean ) : Boolean
13788: Function jxWin32MoveFileReplaceExisting( const SrcFileName, DstFileName : string ) : Boolean
13789: Function jxWin32BackupFile( const FileName : string; Move : Boolean ) : Boolean
13790: Function jxWin32RestoreFile( const FileName : string ) : Boolean
13791: Function jxSamePath( const Path1, Path2 : string ) : Boolean
13792: Procedure jxPathListAddItems( var List : string; const Items : string )
13793: Procedure jxPathListIncludeItems( var List : string; const Items : string )
13794: Procedure jxPathListDelItems( var List : string; const Items : string )
13795: Procedure jxPathListDelItem( var List : string; const Index : Integer )
13796: Function jxPathListItemCount( const List : string ) : Integer
13797: Function jxPathListGetItem( const List : string; const Index : Integer ) : string
13798: Procedure jxPathListSetItem( var List : string; const Index : Integer; const Value : string )
13799: Function jxPathListItemIndex( const List, Item : string ) : Integer
13800: Function jxParamName( Idx:Int;const Separator:string;const AllowedPrefixChars:string;TrimName:Bool ):string;
13801: Function jxParamValue( Index : Integer; const Separator : string; TrimValue : Boolean ) : string;
13802: Function jxParamValue2( const SearchName:string; const Separator : string; CaseSensitive : Boolean; const
    AllowedPrefixCharacters : string; TrimValue : Boolean ) : string;
13803: Function jxParamPos( const SearchName : string; const Separator : string; CaseSensitive : Boolean; const
    AllowedPrefixCharacters : string ) : Integer
13804: end;
13805:
13806: procedure SIRegister_FileUtil(CL: TPSPascalCompiler);
13807: begin
13808:   'UTF8FileHeader', 'String #$ef#$bb#$bf';
13809:   Function lCompareFilenames( const Filenamel, Filename2 : string ) : integer
13810:   Function lCompareFilenamesIgnoreCase( const Filenamel, Filename2 : string ) : integer
13811:   Function lCompareFilenames( const Filenamel, Filename2 : string; ResolveLinks : boolean ) : integer
13812:   Function lCompareFilenames(Filenamel:PChar;Len1:int;Filename2:PChar;Len2:int;ResolveLinks:boolean):int;
13813:   Function lFilenameIsAbsolute( const TheFilename : string ) : boolean
13814:   Function lFilenameIsWinAbsolute( const Thefilename : string ) : boolean
13815:   Function lFilenameIsUnixAbsolute( const Thefilename : string ) : boolean
13816:   Procedure lCheckIfFileIsExecutable( const AFilename : string )
13817:   Procedure lCheckIfFileIsSymlink( const AFilename : string )
13818:   Function lFileIsReadable( const AFilename : string ) : boolean
13819:   Function lFileIsWritable( const AFilename : string ) : boolean
13820:   Function lFileIsText( const AFilename : string; out FileReadable : boolean ) : boolean
13821:   Function lFileIsText( const AFilename : string ) : boolean
13822:   Function lFileIsExecutable( const AFilename : string ) : boolean
13823:   Function lFileIsSymlink( const AFilename : string ) : boolean
13824:   Function lFileIsHardLink( const AFilename : string ) : boolean
13825:   Function lFileSize( const Filenamel : string ) : int64;
13826:   Function lGetFileDescription( const AFilename : string ) : string
13827:   Function lReadAllLinks( const Filenamel : string; ExceptionOnErr : boolean ) : string
13828:   Function lTryReadAllLinks( const Filenamel : string ) : string
13829:   Function lDirPathExists( const FileName : String ) : Boolean
13830:   Function lForceDirectory( DirectoryName : string ) : boolean
13831:   Function lDeleteDirectory( const DirectoryName : string; OnlyChildren : boolean ) : boolean
13832:   Function lProgramDirectory : string
13833:   Function lDirectoryIsWritable( const DirectoryName : string ) : boolean
13834:   Function lExtractFileNameOnly( const AFilename : string ) : string
13835:   Function lExtractFileNameWithoutExt( const AFilename : string ) : string
13836:   Function lCompareFileExt( const Filenamel, Ext : string; CaseSensitive : boolean ) : integer;
13837:   Function lCompareFileExt( const Filenamel, Ext : string ) : integer;
13838:   Function lFilenameIsPascalUnit( const Filenamel : string ) : boolean
13839:   Function lAppendPathDelim( const Path : string ) : string
13840:   Function lChompPathDelim( const Path : string ) : string
13841:   Function lTrimFilename( const AFilename : string ) : string
13842:   Function lCleanAndExpandFilename( const Filenamel : string ) : string
13843:   Function lCleanAndExpandDirectory( const Filenamel : string ) : string
13844:   Function lCreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string ) : string
13845:   Function lCreateRelativePath( const Filenamel, BaseDirectory : string; UsePointDirectory : boolean;
    AlwaysRequireSharedBaseFolder : Boolean ) : string
13846:   Function lCreateAbsolutePath( const Filenamel, BaseDirectory : string ) : string
13847:   Function lFileIsInPath( const Filenamel, Path : string ) : boolean
13848:   Function lFileIsInDirectory( const Filenamel, Directory : string ) : boolean
13849:   TSearchFileInPathFlag', '( sffDontSearchInbasePath, sffSearchLoUpCase )
13850:   TSearchFileInPathFlags', 'set of TSearchFileInPathFlag
13851:   'AllDirectoryEntriesMask', 'String '*
13852:   Function l GetAllFilesMask : string
13853:   Function lGetExeExt : string
13854:   Function lSearchFileInPath( const Filenamel, basePath, SearchPath, Delimiter : string; Flags :
    TSearchFileInPathFlags ) : string
13855:   Function lSearchAllFilesInPath( const Filenamel, basePath, SearchPath, Delimiter:string;Flags :
    TSearchFileInPathFlags ) : TStrings
13856:   Function lFindDiskFilename( const Filenamel : string ) : string
13857:   Function lFindDiskFileCaseInsensitive( const Filenamel : string ) : string
13858:   Function lFindDefaultExecutablePath( const Executable : string; const BaseDir: string):string
13859:   Function lGetDarwinSystemFilename( Filenamel : string ) : string
13860:   SIRegister_TFileIterator(CL);
13861:   TFileFoundEvent', 'Procedure ( FileIterator : TFileIterator )
13862:   TDirectoryFoundEvent', 'Procedure ( FileIterator : TFileIterator )
13863:   TDirectoryEnterEvent', 'Procedure ( FileIterator : TFileIterator )
13864:   SIRegister_TFileSearcher(CL);
13865:   Function lFindAllFiles(const SearchPath:String;SearchMsk:String;SearchSubDirs:Bool):TStringList
13866:   Function lFindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean ) : TStringList
13867:   // TCopyFileFlag', '( cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime )
13868:   // TCopyFileFlags', 'set of TCopyFileFlag
13869:   Function lCopyFile( const SrcFilename, DestFilename : string; Flags : TCopyFileFlags ) : boolean
13870:   Function lCopyFile( const SrcFilename, DestFilename : string; PreserveTime : boolean) : boolean

```

```

13871: Function lCopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
13872: Function lReadFileToString( const Filename : string) : string
13873: Function lGetTempFilename( const Directory, Prefix : string) : string
13874: {Function NeedRTLAnsi : boolean
13875: Procedure SetNeedRTLAnsi( NewValue : boolean)
13876: Function UTF8ToSys( const s : string) : string
13877: Function SysToUTF8( const s : string) : string
13878: Function ConsoleToUTF8( const s : string) : string
13879: Function UTF8ToConsole( const s : string) : string}
13880: Function FileExistsUTF8( const FileName : string) : boolean
13881: Function FileAgeUTF8( const FileName : string) : Longint
13882: Function DirectoryExistsUTF8( const Directory : string) : Boolean
13883: Function ExpandFileNameUTF8( const FileName : string) : string
13884: Function ExpandUNCfileNameUTF8( const FileName : string) : string
13885: Function ExtractShortPathNameUTF8( const FileName : String) : String
13886: Function FindFirstUTF8(const Path: string; Attr : Longint; out Rslt : TSearchRec) : Longint
13887: Function FindNextUTF8( var Rslt : TSearchRec) : Longint
13888: Procedure FindCloseUTF8( var F : TSearchrec)
13889: Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
13890: Function FileGetAttrUTF8( const FileName : String) : Longint
13891: Function FileSetAttrUTF8( const FileName : String; Attr : longint) : Longint
13892: Function DeleteFileUTF8( const FileName : String) : Boolean
13893: Function RenameFileUTF8( const OldName, NewName : String) : Boolean
13894: Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
13895: Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
13896: Function GetCurrentDirUTF8 : String
13897: Function SetCurrentDirUTF8( const NewDir : String) : Boolean
13898: Function CreateDirUTF8( const NewDir : String) : Boolean
13899: Function RemoveDirUTF8( const Dir : String) : Boolean
13900: Function ForceDirectoriesUTF8( const Dir : string) : Boolean
13901: Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
13902: Function FileCreateUTF8( const FileName : string) : THandle;
13903: Function FileCreateUTF8l( const FileName : string; Rights : Cardinal) : THandle;
13904: Function ParamStrUTF8( Param : Integer) : string
13905: Function GetEnvironmentStringUTF8( Index : Integer) : string
13906: Function GetEnvironmentVariableUTF8( const EnvVar : string) : String
13907: Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
13908: Function GetAppConfigFileUTF8(Global:Boolean; SubDir:boolean; CreateDir : boolean) : string
13909: Function SysErrorMessageUTF8( ErrorCode : Integer) : String
13910: end;
13911:
13912: procedure SIRegister_Keyboard(CL: TPSPascalCompiler);
13913: begin
13914: //VK_F23 = 134;
13915: //{$EXTERNALSYM VK_F24}
13916: //VK_F24 = 135;
13917: TVirtualKeyCode', 'Integer
13918: 'VK_MOUSEWHEELUP','integer'(134);
13919: 'VK_MOUSEWHEELDOWN','integer'(135);
13920: Function glIsKeyDown( c : Char ) : Boolean;
13921: Function glIsKeyDown( vk : TVirtualKeyCode ) : Boolean;
13922: Function glKeyPressed( minVkCode : TVirtualKeyCode ) : TVirtualKeyCode
13923: Function glVirtualKeyCodeToKeyName( vk : TVirtualKeyCode ) : String
13924: Function glKeyNameToVirtualKeyCode( const keyName : String ) : TVirtualKeyCode
13925: Function glCharToVirtualKeyCode( c : Char ) : TVirtualKeyCode
13926: Procedure glKeyboardNotifyWheelMoved( wheelDelta : Integer )
13927: end;
13928:
13929: procedure SIRegister_GLCrossPlatform(CL: TPSPascalCompiler);
13930: begin
13931: TGLPoint', 'TPoint
13932: //PGLPoint', '^TGLPoint // will not work
13933: TGLRect', 'TRect
13934: //PGLRect', '^TGLRect // will not work
13935: TDelphiColor', 'TColor
13936: TGLPicture', 'TPicture
13937: TGLGraphic', 'TGraphic
13938: TGLBitmap', 'TBitmap
13939: //TGraphicClass', 'class of TGraphic
13940: TGLTextLayout', '( tlTop, tlCenter, tlBottom )
13941: TGLMouseEventButton', '( mbLeft, mbRight, mbMiddle )
13942: TGLMouseEvent', 'Procedure ( Sender : TObject; Button : TGLMouse'
13943: +'Button; Shift : TShiftState; X, Y : Integer)
13944: TGLMouseEventEvent', 'TMouseEvent
13945: TGLKeyEvent', 'TKeyEvent
13946: TGLKeyPressEvent', 'TKeyPressEvent
13947: EGLOSError', 'EWin32Error
13948: EGLOSError', 'EWin32Error
13949: EGLOSError', 'EOSError
13950: 'glsAllFilter', 'string'All // $AllFilter
13951: Function GLPoint( const x, y : Integer ) : TGLPoint
13952: Function GLRGB( const r, g, b : Byte ) : TColor
13953: Function GLColorToRGB( color : TColor ) : TColor
13954: Function GLGetRValue( rgb : DWORD ) : Byte
13955: Function GLGetGValue( rgb : DWORD ) : Byte
13956: Function GLGetBValue( rgb : DWORD ) : Byte
13957: Procedure GLInitWinColors
13958: Function GLRect( const aLeft, aTop, aRight, aBottom : Integer ) : TGLRect
13959: Procedure GLInflateGLRect( var aRect : TGLRect; dx, dy : Integer )

```

```

13960: Procedure GLIntersectGLRect( var aRect : TGLRect; const rect2 : TGLRect )
13961: Procedure GLInformationDlg( const msg : String )
13962: Function GLQuestionDlg( const msg : String ) : Boolean
13963: Function GLInputDlg( const aCaption, aPrompt, aDefault : String ) : String
13964: Function GLSavePictureDialog( var aFileName : String; const aTitle : String ) : Boolean
13965: Function GLOpenPictureDialog( var aFileName : String; const aTitle : String ) : Boolean
13966: Function GLApplicationTerminated : Boolean
13967: Procedure GLRaiseLastOSError
13968: Procedure GLFreeAndNil( var anObject: TObject )
13969: Function GLGetDeviceLogicalPixelsX( device : Cardinal ) : Integer
13970: Function GLGetCurrentColorDepth : Integer
13971: Function GLPixelFormatToColorBits( aPixelFormat : TPixelFormat ) : Integer
13972: Function GLBitmapScanLine( aBitmap : TGLBitmap; aRow : Integer ) : Pointer
13973: Procedure GLSleep( length : Cardinal )
13974: Procedure GLQueryPerformanceCounter( var val : Int64 )
13975: Function GLQueryPerformanceFrequency( var val : Int64 ) : Boolean
13976: Function GLStartPrecisionTimer : Int64
13977: Function GLPrecisionTimerLap( const precisionTimer : Int64 ) : Double
13978: Function GLStopPrecisionTimer( const precisionTimer : Int64 ) : Double
13979: Function GLRDTSC : Int64
13980: Procedure GLLoadBitmapFromInstance( ABitmap : TBitmap; AName : string )
13981: Function GLOKMessageBox( const Text, Caption : string ) : Integer
13982: Procedure GLShowHTMLUrl( Url : String )
13983: Procedure GLShowCursor( AShow : boolean )
13984: Procedure GLSetCursorPos( AScreenX, AScreenY : integer )
13985: Procedure GLGetCursorPos( var point : TGLPoint )
13986: Function GLGetScreenWidth : integer
13987: Function GLGetScreenHeight : integer
13988: Function GLGetTickCount : int64
13989: function RemoveSpaces(const str : String) : String;
13990:   TNoramlMapSpace'.'( nmsObject, nmsTangent )
13991: Procedure CalcObjectSpaceLightVectors(Light:TAffineVector;Vertices TAffineVectorList:Colors:TVectorList)
13992: Procedure SetupTangentSpace( Vertices, Normals, TexCoords, Tangents, BiNormals : TAffineVectorList )
13993: Procedure CalcTangentSpaceLightVectors( Light : TAffineVector; Vertices, Normals, Tangents, BiNormals : TAffineVectorList; Colors : TVectorList )
13994: Function CreateObjectSpaceNormalMap( Width,Height:Integer;HiNormals,
HiTexCoords:TAffineVectorList):TGLBitmap
13995: Function CreateTangentSpaceNormalMap( Width, Height : Integer; HiNormals, HiTexCoords, LoNormals,
LoTexCoords, Tangents, BiNormals : TAffineVectorList ) : TGLBitmap
13996: end;
13997:
13998: procedure SIRegister_GLStarRecord(CL: TPSPascalCompiler);
13999: begin
14000:   TGLStarRecord', 'record RA: Word; DEC: SmallInt; BVColorIndex: Byte; VMagnitude: Byte; end
14001: // PGLStarRecord', '^TGLStarRecord // will not work
14002: Function StarRecordPositionZUp( const starRecord : TGLStarRecord ) : TAffineVector
14003: Function StarRecordPositionYUp( const starRecord : TGLStarRecord ) : TAffineVector
14004: Function StarRecordColor( const starRecord : TGLStarRecord; bias : Single ) : TVector
14005: end;
14006:
14007:
14008: procedure SIRegister_GeometryBB(CL: TPSPascalCompiler);
14009: begin
14010:   TAABB', 'record min : TAffineVector; max : TAffineVector; end
14011: //PAABB', '^TAABB // will not work
14012: TBSphere', 'record Center : TAffineVector; Radius : single; end
14013: TClipRect', 'record Left : Single; Top:Single; Right:Single; Bottom : Single; end
14014: TSpaceContains', '(scNoOverlap, scContainsFully, scContainsPartially )
14015: Function AddBB( var c1 : THmgBoundingBox; const c2 : THmgBoundingBox ) : THmgBoundingBox
14016: Procedure AddAABB( var aabb : TAABB; const aabb1 : TAABB )
14017: Procedure SetBB( var c : THmgBoundingBox; const v : TVector )
14018: Procedure SetAABB( var bb : TAABB; const v : TVector )
14019: Procedure BBTransform( var c : THmgBoundingBox; const m : TMatrix )
14020: Procedure AABBTTransform( var bb : TAABB; const m : TMatrix )
14021: Procedure AABBScale( var bb : TAABB; const v : TAffineVector )
14022: Function BBMinX( const c : THmgBoundingBox ) : Single
14023: Function BBMaxX( const c : THmgBoundingBox ) : Single
14024: Function BBMinY( const c : THmgBoundingBox ) : Single
14025: Function BBMaxY( const c : THmgBoundingBox ) : Single
14026: Function BBMinZ( const c : THmgBoundingBox ) : Single
14027: Function BBMaxZ( const c : THmgBoundingBox ) : Single
14028: Procedure AABBIInclude( var bb : TAABB; const p : TAffineVector )
14029: Procedure AABBFfromSweep( var SweepAABB : TAABB; const Start, Dest : TVector; const Radius : Single )
14030: Function AABBIntersection( const aabb1, aabb2 : TAABB ) : TAABB
14031: Function BBTAAABB( const aBB : THmgBoundingBox ) : TAABB
14032: Function AABBTtoBB( const anAABB : TAABB ) : THmgBoundingBox;
14033: Function AABBTtoBB1( const anAABB : TAABB; const m : TMatrix ) : THmgBoundingBox;
14034: Procedure OffsetAABB( var aabb : TAABB; const delta : TAffineVector );
14035: Procedure OffsetAABB1( var aabb : TAABB; const delta : TVector );
14036: Function IntersectAABBS( const aabb1, aabb2 : TAABB; const m1To2, m2To1 : TMatrix ) : Boolean;
14037: Function IntersectAABBSAbsoluteXY( const aabb1, aabb2 : TAABB ) : Boolean
14038: Function IntersectAABBSAbsoluteXZ( const aabb1, aabb2 : TAABB ) : Boolean
14039: Function IntersectAABBSAbsolute( const aabb1, aabb2 : TAABB ) : Boolean
14040: Function AABBfitsInAABBAbsolute( const aabb1, aabb2 : TAABB ) : Boolean
14041: Function PointInAABB( const p : TAffineVector; const aabb : TAABB ) : Boolean;
14042: Function PointInAABB1( const p : TVector; const aabb : TAABB ) : Boolean;
14043: Function PlaneIntersectAABB( Normal : TAffineVector; d : single; aabb : TAABB ) : boolean
14044: Function TriangleIntersectAABB( const aabb : TAABB; v1, v2, v3 : TAffineVector ) : boolean
14045: Procedure ExtractAABBCorners( const AABB : TAABB; var AABBCorners : TAABBCorners )

```

```

14046: Procedure AABBTosSphere( const AABB : TAABB; var BSphere : TBSphere);
14047: Procedure BSphereToAABB( const BSphere : TBSphere; var AABB : TAABB);
14048: Function BSphereToAABB1( const center : TAffineVector; radius : Single ) : TAABB;
14049: Function BSphereToAABB2( const center : TVector; radius : Single ) : TAABB;
14050: Function AABBContainsAABB( const mainAABB, testAABB : TAABB ) : TSpaceContains;
14051: Function BSphereContainsAABB( const mainBSphere : TBSphere; const testAABB : TAABB ) : TSpaceContains;
14052: Function BSphereContainsBSphere( const mainBSphere, testBSphere : TBSphere ) : TSpaceContains;
14053: Function AABBContainsBSphere( const mainAABB : TAABB; const testBSphere : TBSphere ) : TSpaceContains;
14054: Function PlaneContainsBSphere( const Location,Normal:TAffineVector;const
testBSphere:TBSphere):TSpaceContains;
14055: Function FrustumContainsBSphere( const Frustum : TFrustum; const testBSphere : TBSphere ) : TSpaceContains;
14056: Function FrustumContainsAABB( const Frustum : TFrustum; const testAABB : TAABB ) : TSpaceContains;
14057: Function ClipToAABB( const v : TAffineVector; const AABB : TAABB ) : TAffineVector;
14058: Function BSphereIntersectsBSphere( const mainBSphere, testBSphere : TBSphere ) : boolean;
14059: Procedure IncludeInClipRect( var clipRect : TClipRect; x, y : Single );
14060: Function AABBToClipRect( const aabb:TAABB;modelViewProjection:TMATRIX;viewportSizeX,
viewportSizeY:Int ):TClipRect;
14061: end;
14062:
14063: procedure SIRegister_GeometryCoordinates(CL: TPSPPascalCompiler);
14064: begin
14065:   Procedure Cylindrical_Cartesian( const r, theta, z1 : single; var x, y, z : single );
14066:   Procedure Cylindrical_Cartesian1( const r, theta, z1 : double; var x, y, z : double );
14067:   Procedure Cylindrical_Cartesian2( const r,theta,z1 : single; var x, y, z : single; var ierr : integer );
14068:   Procedure Cylindrical_Cartesian3( const r,theta,z1 : double; var x, y, z : double; var ierr : integer );
14069:   Procedure Cartesian_Cylindrical( const x, y, z1 : single; var r, theta, z : single );
14070:   Procedure Cartesian_Cylindrical1( const x, y, z1 : double; var r, theta, z : double );
14071:   Procedure Spherical_Cartesian( const r, theta, phi : single; var x, y, z : single );
14072:   Procedure Spherical_Cartesian1( const r, theta, phi : double; var x, y, z : double );
14073:   Procedure Spherical_Cartesian2( const r,theta, phi : single; var x, y, z : single; var ierr : integer );
14074:   Procedure Spherical_Cartesian3( const r,theta, phi : double; var x, y, z : double; var ierr : integer );
14075:   Procedure Cartesian_Spherical( const x, y, z : single; var r, theta, phi : single );
14076:   Procedure Cartesian_Spherical1( const v : TAffineVector; var r, theta, phi : Single );
14077:   Procedure Cartesian_Spherical2( const x, y, z : double; var r, theta, phi : double );
14078:   Procedure ProlateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single );
14079:   Procedure ProlateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double );
14080:   Procedure ProlateSpheroidal_Cartesian2( const xi,eta,phi,a:single;var x,y,z:single;var ierr:integer );
14081:   Procedure ProlateSpheroidal_Cartesian3( const xi,eta,phi,a:double;var x,y,z:double;var ierr:integer );
14082:   Procedure OblateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single );
14083:   Procedure OblateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double );
14084:   Procedure OblateSpheroidal_Cartesian2( const xi,eta,phi,a:single;var x,y,z:single;var ierr:integer );
14085:   Procedure OblateSpheroidal_Cartesian3( const xi,eta,phi,a:double;var x,y,z:double;var ierr:integer );
14086:   Procedure BipolarCylindrical_Cartesian( const u, v, z1, a : single; var x, y, z : single );
14087:   Procedure BipolarCylindrical_Cartesian1( const u, v, z1, a : double; var x, y, z : double );
14088:   Procedure BipolarCylindrical_Cartesian2( const u,v,z1,a: single;var x,y,z:single; var ierr : integer );
14089:   Procedure BipolarCylindrical_Cartesian3( const u,v,z1,a: double;var x,y,z:double; var ierr : integer );
14090: end;
14091:
14092: procedure SIRegister_VectorGeometry(CL: TPSPPascalCompiler);
14093: begin
14094:   'EPSILON','Single').setExtended( 1e-40 );
14095:   'EPSILON2','Single').setExtended( 1e-30 );  }
14096: TRenderContextClippingInfo', 'record origin : TVector; clippingD'
14097:   +'irection : TVector; viewPortRadius : Single; farClippingDistance:Single;frustum:TFrustum; end
14098: THmgPlane', 'TVector
14099: TDoublHmgPlane', 'THomogeneousDblVector
14100: {TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
14101:   +'XZ, ttShearyZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
14102:   +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW ) }
14103: TSinglArray', 'array of Single
14104: TTransformations', 'array [0..15] of Single)
14105: TPackedRotationMatrix', 'array [0..2] of Smallint)
14106: TVertex', 'TAffineVector
14107: //TVectorGL', 'THomogeneousFltVector
14108: //TMATRIXGL', 'THomogeneousFltMatrix
14109: // TPackedRotationMatrix = array [0..2] of SmallInt;
14110: Function glTexPointMake( const s, t : Single ) : TTExPoint
14111: Function glAffineVectorMake( const x, y, z : Single ) : TAffineVector;
14112: Function glAffineVectorMakel( const v : TVectorGL ) : TAffineVector;
14113: Procedure glSetAffineVector( var v : TAffineVector; const x, y, z : Single );
14114: Procedure glSetVector( var v : TAffineVector; const x, y, z : Single );
14115: Procedure glSetVector1( var v : TAffineVector; const vSrc : TVectorGL );
14116: Procedure glSetVector2( var v : TAffineVector; const vSrc : TAffineVector );
14117: Procedure glSetVector3( var v : TAffineDblVector; const vSrc : TAffineVector );
14118: Procedure glSetVector4( var v : TAffineDblVector; const vSrc : TVectorGL );
14119: Function glVectorMake( const v : TAffineVector; w : Single ) : TVectorGL;
14120: Function glVectorMakel( const x, y, z : Single; w : Single ) : TVectorGL;
14121: Function glPointMake( const x, y, z : Single ) : TVectorGL;
14122: Function glPointMakel( const v : TAffineVector ) : TVectorGL;
14123: Function glPointMake2( const v : TVectorGL ) : TVectorGL;
14124: Procedure glSetVector5( var v : TVectorGL; const x, y, z : Single; w : Single );
14125: Procedure glSetVector6( var v : TVectorGL; const av : TAffineVector; w : Single );
14126: Procedure glg1SetVector7( var v : TVectorGL; const vSrc : TVectorGL );
14127: Procedure glMakePoint( var v : TVectorGL; const x, y, z : Single );
14128: Procedure glMakePoint1( var v : TVectorGL; const av : TAffineVector );
14129: Procedure glMakePoint2( var v : TVectorGL; const av : TVectorGL );
14130: Procedure glMakeVector( var v : TAffineVector; const x, y, z : Single );
14131: Procedure glMakeVector1( var v : TVectorGL; const x, y, z : Single );
14132: Procedure glMakeVector2( var v : TVectorGL; const av : TAffineVector );

```

```

14133: Procedure glMakeVector3( var v : TVectorGL; const av : TVectorGL);
14134: Procedure glRstVector( var v : TAffineVector);
14135: Procedure glRstVector1( var v : TVectorGL);
14136: Function glVectorAdd( const v1, v2 : TAffineVector ) : TAffineVector;
14137: Procedure glVectorAdd1( const v1, v2 : TAffineVector; var vr : TAffineVector );
14138: //Procedure VectorAdd2( const v1, v2 : TAffineVector; vr : PAffineVector );
14139: Function glVectorAdd3( const v1, v2 : TVectorGL ) : TVectorGL;
14140: Procedure glVectorAdd4( const v1, v2 : TVectorGL; var vr : TVectorGL );
14141: Function glVectorAdd5( const v : TAffineVector; const f : Single ) : TAffineVector;
14142: Function glVectorAdd6( const v : TVectorGL; const f : Single ) : TVectorGL;
14143: Procedure glAddVector7( var v1 : TAffineVector; const v2 : TAffineVector );
14144: Procedure glAddVector8( var v1 : TAffineVector; const v2 : TVectorGL );
14145: Procedure glAddVector9( var v1 : TVectorGL; const v2 : TVectorGL );
14146: Procedure glAddVector10( var v : TAffineVector; const f : Single );
14147: Procedure glAddVector11( var v : TVectorGL; const f : Single );
14148: //Procedure TexPointArrayAdd(const src:PTexPointArray,const delta:TTexPoint,const
nb:Int;dest:PTexPointArray);
14149: //Procedure TexPointArrayScaleAndAdd(const src:PTexPointArray,const delta:TTexPoint,const
nb:Integer;const scale: TTExPoint; dest : PTExPointArray);
14150: //Procedure VectorArrayAdd(const src:PAffineVectorArray,const delta:TAffineVector,const nb:Integer;dest:
PAffineVectorArray);
14151: Function glVectorSubtract( const V1, V2 : TAffineVector ) : TAffineVector;
14152: Procedure glVectorSubtract1( const v1, v2 : TAffineVector; var result : TAffineVector );
14153: Procedure glVectorSubtract2( const v1, v2 : TAffineVector; var result : TVectorGL );
14154: Procedure glVectorSubtract3( const v1, v2 : TVectorGL; var result : TVectorGL );
14155: Function glVectorSubtract4( const V1, V2 : TVectorGL ) : TVectorGL;
14156: Procedure glVectorSubtract5( const v1, v2 : TVectorGL; var result : TVectorGL );
14157: Procedure glVectorSubtract6( const v1, v2 : TVectorGL; var result : TAffineVector );
14158: Function glVectorSubtract7( const v1 : TAffineVector; delta : Single ) : TAffineVector;
14159: Function glVectorSubtract8( const v1 : TVectorGL; delta : Single ) : TVectorGL;
14160: Procedure glSubtractVector9( var V1 : TAffineVector; const V2 : TAffineVector );
14161: Procedure glSubtractVector10( var V1 : TVectorGL; const V2 : TVectorGL );
14162: Procedure glCombineVector( var vr : TAffineVector; const v : TAffineVector; var f : Single );
14163: //Procedure CombineVector1( var vr : TAffineVector; const v : TAffineVector; pf : PFloat );
14164: Function glTexPointCombine( const t1, t2 : TTExPoint; f1, f2 : Single ) : TTExPoint;
14165: Function glVectorCombine2( const V1, V2 : TAffineVector; const F1, F2 : Single ) : TAffineVector;
14166: Function glVectorCombine33( const V1, V2, V3 : TAffineVector; const F1, F2, F3 : Single ) : TAffineVector;
14167: Procedure glVectorCombine34( const V1,V2,V3:TAffineVector; const F1,F2,F3: Single; var vr : TAffineVector );
14168: Procedure glCombineVector5( var vr : TVectorGL; const v : TVectorGL; var f : Single );
14169: Procedure glCombineVector6( var vr : TVectorGL; const v : TAffineVector; var f : Single );
14170: Function glVectorCombine7( const V1, V2 : TVectorGL; const F1, F2 : Single ) : TVectorGL;
14171: Function glVectorCombine8( const V1 : TVectorGL; const V2 : TAffineVector; const F1, F2:Single ) : TVectorGL;
14172: Procedure glVectorCombine9( const V1 : TVectorGL; const V2 : TAffineVector; const F1, F2:Single; var vr : TVectorGL );
14173: Procedure glVectorCombine10( const V1, V2 : TVectorGL; const F1, F2 : Single; var vr : TVectorGL );
14174: Procedure glVectorCombine11( const V1, V2 : TVectorGL; const F2 : Single; var vr : TVectorGL );
14175: Function glVectorCombine3( const V1, V2, V3 : TVectorGL; const F1, F2, F3 : Single ) : TVectorGL;
14176: Procedure glVectorCombine31( const V1, V2, V3 : TVectorGL; const F1, F2, F3:Single; var vr : TVectorGL );
14177: Function glVectorDotProduct( const V1, V2 : TAffineVector ) : Single;
14178: Function glVectorDotProduct1( const V1, V2 : TVectorGL ) : Single;
14179: Function glVectorDotProduct2( const V1 : TVectorGL; const V2 : TAffineVector ) : Single;
14180: Function glPointProject( const p, origin, direction : TAffineVector ) : Single;
14181: Function glPointProject1( const p, origin, direction : TVectorGL ) : Single;
14182: Function glVectorCrossProduct( const V1, V2 : TAffineVector ) : TAffineVector;
14183: Function glVectorCrossProduct1( const V1, V2 : TVectorGL ) : TVectorGL;
14184: Procedure glVectorCrossProduct2( const v1, v2 : TVectorGL; var vr : TVectorGL );
14185: Procedure glVectorCrossProduct3( const v1, v2 : TAffineVector; var vr : TVectorGL );
14186: Procedure glVectorCrossProduct4( const v1, v2 : TVectorGL; var vr : TAffineVector );
14187: Procedure glVectorCrossProduct5( const v1, v2 : TAffineVector; var vr : TAffineVector );
14188: Function glLerp( const start, stop, t : Single ) : Single;
14189: Function glAngleLerp( start, stop, t : Single ) : Single;
14190: Function glDistanceBetweenAngles( angle1, angle2 : Single ) : Single;
14191: Function glTexPointLerp( const t1, t2 : TTExPoint; t : Single ) : TTExPoint;
14192: Function glVectorLerp( const v1, v2 : TAffineVector; t : Single ) : TAffineVector;
14193: Procedure glVectorLerp1( const v1, v2 : TAffineVector; t : Single; var vr : TAffineVector );
14194: Function glVectorLerp2( const v1, v2 : TVectorGL; t : Single ) : TVectorGL;
14195: Procedure glVectorLerp3( const v1, v2 : TVectorGL; t : Single; var vr : TVectorGL );
14196: Function glVectorAngleLerp( const v1, v2 : TAffineVector; t : Single ) : TAffineVector;
14197: Function glVectorAngleCombine( const v1, v2 : TAffineVector; f : Single ) : TAffineVector;
14198: // Procedure VectorArrayLerp( const src1, src2:PVectorArray; t:Single; n:Integer; dest:PVectorArray );
14199: // Procedure VectorArrayLerp1( const src1, src2 : PAffineVectorArray; t : Single; n : Integer; dest :
PAffineVectorArray );
14200: Function glVectorLength( const x, y : Single ) : Single;
14201: Function glVectorLength1( const x, y, z : Single ) : Single;
14202: Function glVectorLength2( const v : TAffineVector ) : Single;
14203: Function glVectorLength3( const v : TVectorGL ) : Single;
14204: Function glVectorlength4( const v : array of Single ) : Single;
14205: Function glVectorNorm( const x, y : Single ) : Single;
14206: Function glVectorNorm1( const v : TAffineVector ) : Single;
14207: Function glVectorNorm2( const v : TVectorGL ) : Single;
14208: Function glVectorNorm3( var V : array of Single ) : Single;
14209: Procedure glNormalizeVector( var v : TAffineVector );
14210: Procedure glNormalizeVector1( var v : TVectorGL );
14211: Function glVectorNormalize( const v : TAffineVector ) : TAffineVector;
14212: Function glVectorNormalize1( const v : TVectorGL ) : TVectorGL;
14213: // Procedure NormalizeVectorArray( list : PAffineVectorArray; n : Integer );
14214: Function glVectorAngleCosine( const V1, V2 : TAffineVector ) : Single;
14215: Function glVectorNegate( const v : TAffineVector ) : TAffineVector;
14216: Function glVectorNegate1( const v : TVectorGL ) : TVectorGL;
14217: Procedure glNegateVector( var V : TAffineVector );

```

```

14218: Procedure glNegateVector2( var V : TVectorGL);
14219: Procedure glNegateVector3( var V : array of Single);
14220: Procedure glScaleVector( var v : TAffineVector; factor : Single);
14221: Procedure glScaleVector1( var v : TAffineVector; const factor : TAffineVector);
14222: Procedure glScaleVector2( var v : TVectorGL; factor : Single);
14223: Procedure glScaleVector3( var v : TVectorGL; const factor : TVectorGL);
14224: Function glVectorScale( const v : TAffineVector; factor : Single) : TAffineVector;
14225: Procedure glVectorScale1( const v : TAffineVector; factor : Single; var vr : TAffineVector);
14226: Function glVectorScale2( const v : TVectorGL; factor : Single) : TVectorGL;
14227: Procedure glVectorScale3( const v : TVectorGL; factor : Single; var vr : TVectorGL);
14228: Procedure glVectorScale4( const v : TVectorGL; factor : Single; var vr : TAffineVector);
14229: Procedure glDivideVector( var v : TVectorGL; const divider : TVectorGL);
14230: Function glVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14231: Function glVectorEquals1( const V1, V2 : TAffineVector) : Boolean;
14232: Function glAffineVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14233: Function glVectorIsNull( const v : TVectorGL) : Boolean;
14234: Function glVectorIsNotNull( const v : TAffineVector) : Boolean;
14235: Function glVectorSpacing( const v1, v2 : TTExPoint) : Single;
14236: Function glVectorSpacing1( const v1, v2 : TAffineVector) : Single;
14237: Function glVectorSpacing2( const v1, v2 : TVectorGL) : Single;
14238: Function glVectorDistance( const v1, v2 : TAffineVector) : Single;
14239: Function glVectorDistance1( const v1, v2 : TVectorGL) : Single;
14240: Function glVectorDistance2( const v1, v2 : TAffineVector) : Single;
14241: Function glVectorDistance21( const v1, v2 : TVectorGL) : Single;
14242: Function glVectorPerpendicular( const V, N : TAffineVector) : TAffineVector;
14243: Function glVectorReflect( const V, N : TAffineVector) : TAffineVector;
14244: Procedure glRotateVector( var vector : TVectorGL; const axis : TAffineVector; angle : Single);
14245: Procedure glRotateVector1( var vector : TVectorGL; const axis : TVectorGL; angle : Single);
14246: Procedure glRotateVectorAroundY( var v : TAffineVector; alpha : Single);
14247: Function glVectorRotateAroundX( const v : TAffineVector; alpha : Single) : TAffineVector;
14248: Function glVectorRotateAroundY( const v : TAffineVector; alpha : Single) : TAffineVector;
14249: Procedure glVectorRotateAroundY1( const v : TAffineVector; alpha : Single; var vr : TAffineVector);
14250: Function glVectorRotateAroundZ( const v : TAffineVector; alpha : Single) : TAffineVector;
14251: Procedure glAbsVector( var v : TVectorGL);
14252: Procedure glAbsVector1( var v : TAffineVector);
14253: Function glVectorAbs( const v : TVectorGL) : TVectorGL;
14254: Function glVectorAbs1( const v : TAffineVector) : TAffineVector;
14255: Procedure glSetMatrix( var dest : THomogeneousDblMatrix; const src : TMatrixGL);
14256: Procedure glSetMatrix1( var dest : TAffineMatrix; const src : TMatrixGL);
14257: Procedure glSetMatrix2( var dest : TMatrixGL; const src : TAffineMatrix);
14258: Procedure glSetMatrixRow( var dest : TMatrixGL; rowNb : Integer; const aRow : TVectorGL);
14259: Function glCreateScaleMatrix( const v : TAffineVector) : TMatrixGL;
14260: Function glCreateScaleMatrix1( const v : TVectorGL) : TMatrixGL;
14261: Function glCreateTranslationMatrix( const V : TAffineVector) : TMatrixGL;
14262: Function glCreateTranslationMatrix1( const V : TVectorGL) : TMatrixGL;
14263: Function glCreateScaleAndTranslationMatrix( const scale, offset : TVectorGL) : TMatrixGL;
14264: Function glCreateRotationMatrixX( const sine, cosine : Single) : TMatrixGL;
14265: Function glCreateRotationMatrixX1( const angle : Single) : TMatrixGL;
14266: Function glCreateRotationMatrixY( const sine, cosine : Single) : TMatrixGL;
14267: Function glCreateRotationMatrixY1( const angle : Single) : TMatrixGL;
14268: Function glCreateRotationMatrixZ( const sine, cosine : Single) : TMatrixGL;
14269: Function glCreateRotationMatrixZ1( const angle : Single) : TMatrixGL;
14270: Function glCreateRotationMatrix( const anAxis : TAffineVector; angle : Single) : TMatrixGL;
14271: Function glCreateRotationMatrix1( const anAxis : TVectorGL; angle : Single) : TMatrixGL;
14272: Function glCreateAffineRotationMatrix( const anAxis : TAffineVector; angle:Single):TAffineMatrix;
14273: Function glMatrixMultiply( const M1, M2 : TAffineMatrix) : TAffineMatrix;
14274: Function glMatrixMultiply1( const M1, M2 : TMatrixGL) : TMatrixGL;
14275: Procedure glMatrixMultiply2( const M1, M2 : TMatrixGL; var MResult : TMatrixGL);
14276: Function glVectorTransform( const V : TVectorGL; const M : TMatrixGL) : TVectorGL;
14277: Function glVectorTransform1( const V : TVectorGL; const M : TAffineMatrix) : TVectorGL;
14278: Function glVectorTransform2( const V : TAffineVector; const M : TMatrixGL) : TAffineVector;
14279: Function glVectorTransform3( const V : TAffineVector; const M : TAffineMatrix) : TAffineVector;
14280: Function glMatrixDeterminant( const M : TAffineMatrix) : Single;
14281: Function glMatrixDeterminant1( const M : TMatrixGL) : Single;
14282: Procedure glAdjointMatrix( var M : TMatrixGL);
14283: Procedure glAdjointMatrix1( var M : TAffineMatrix);
14284: Procedure glScaleMatrix( var M : TAffineMatrix; const factor : Single);
14285: Procedure glScaleMatrix1( var M : TMatrixGL; const factor : Single);
14286: Procedure glTranslateMatrix( var M : TMatrixGL; const v : TAffineVector);
14287: Procedure glTranslateMatrix1( var M : TMatrixGL; const v : TVectorGL);
14288: Procedure glNormalizeMatrix( var M : TMatrixGL);
14289: Procedure glTransposeMatrix( var M : TAffineMatrix);
14290: Procedure glTransposeMatrix1( var M : TMatrixGL);
14291: Procedure glInvertMatrix( var M : TMatrixGL);
14292: Procedure glInvertMatrix1( var M : TAffineMatrix);
14293: Function glAnglePreservingMatrixInvert( const mat : TMatrixGL) : TMatrixGL;
14294: Function glMatrixDecompose( const M : TMatrixGL; var Tran : TTransformations) : Boolean;
14295: Function glPlaneMake( const p1, p2, p3 : TAffineVector) : THmgPlane;
14296: Function glPlaneMake1( const p1, p2, p3 : TVectorGL) : THmgPlane;
14297: Function glPlaneMake2( const point, normal : TAffineVector) : THmgPlane;
14298: Function glPlaneMake3( const point, normal : TVectorGL) : THmgPlane;
14299: Procedure glSetPlane( var dest : TDoubleHmgPlane; const src : THmgPlane);
14300: Procedure glNormalizePlane( var plane : THmgPlane);
14301: Function glPlaneEvaluatePoint( const plane : THmgPlane; const point : TAffineVector) : Single;
14302: Function glPlaneEvaluatePoint1( const plane : THmgPlane; const point : TVectorGL) : Single;
14303: Function glCalcPlaneNormal( const p1, p2, p3 : TAffineVector) : TAffineVector;
14304: Procedure glCalcPlaneNormal1( const p1, p2, p3 : TAffineVector; var vr : TAffineVector);
14305: Procedure glCalcPlaneNormal2( const p1, p2, p3 : TVectorGL; var vr : TAffineVector);
14306: Function glPointIsInHalfSpace( const point, planePoint, planeNormal : TVectorGL) : Boolean;

```

```

14307: Function glPointIsInHalfSpace( const point, planePoint, planeNormal : TAffineVector) : Boolean;
14308: Function glPointPlaneDistance( const point, planePoint, planeNormal : TVectorGL) : Single;
14309: Function glPointPlaneDistance1( const point, planePoint, planeNormal : TAffineVector) : Single;
14310: Function glPointSegmentClosestPoint( const point, segmentStart, segmentStop:TAffineVector):TAffineVector
14311: Function glPointSegmentDistance( const point, segmentStart, segmentStop : TAffineVector) : single
14312: Function glPointLineClosestPoint( const point, linePoint, lineDirection : TAffineVector) : TAffineVector
14313: Function glPointLineDistance( const point, linePoint, lineDirection : TAffineVector) : Single
14314: Procedure SglementSegmentClosestPoint( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector; var
Segment0Closest, Segment1Closest : TAffineVector)
14315: Function glSegmentSegmentDistance(const S0Start,S0Stop,S1Start,S1Stop:TAffineVector):single
14316: TEulerOrder', (' eulXYZ, eulXZY, eulYXZ, eulYZX, eulZXY, eulZYX)
14317: Function glQuaternionMake( const Imag : array of Single; Real : Single ) : TQuaternion
14318: Function glQuaternionConjugate( const Q : TQuaternion ) : TQuaternion
14319: Function glQuaternionMagnitude( const Q : TQuaternion ) : Single
14320: Procedure glNormalizeQuaternion( var Q : TQuaternion )
14321: Function glQuaternionFromPoints( const V1, V2 : TAffineVector ) : TQuaternion
14322: Procedure glQuaternionToPoints( const Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector )
14323: Function glQuaternionFromMatrix( const mat : TMatrixGL ) : TQuaternion
14324: Function glQuaternionToMatrix( quat : TQuaternion ) : TMatrixGL
14325: Function glQuaternionToAffineMatrix( quat : TQuaternion ) : TAffineMatrix
14326: Function glQuaternionFromAngleAxis( const angle : Single; const axis : TAffineVector ) : TQuaternion
14327: Function glQuaternionFromRollPitchYaw( const r, p, y : Single ) : TQuaternion
14328: Function glQuaternionFromEuler( const x, y, z : Single; eulerOrder : TEulerOrder ) : TQuaternion
14329: Function glQuaternionMultiply( const qL, qR : TQuaternion ) : TQuaternion
14330: Function glQuaternionSlerp( const QStart, QEnd : TQuaternion; Spin : Integer; t : Single ) : TQuaternion;
14331: Function glQuaternionSlerp1( const source, dest : TQuaternion; const t : Single ) : TQuaternion;
14332: Function glLnXPl( X : Extended ) : Extended
14333: Function glLog10( X : Extended ) : Extended
14334: Function glLog2( X : Extended ) : Extended
14335: Function glLog21( X : Single ) : Single;
14336: Function glLogN( Base, X : Extended ) : Extended
14337: Function glIntPower( Base : Extended; Exponent : Integer ) : Extended
14338: Function glPower( const Base, Exponent : Single ) : Single;
14339: Function glPower1( Base : Single; Exponent : Integer ) : Single;
14340: Function glDegToRad( const Degrees : Extended ) : Extended;
14341: Function glDegToRad1( const Degrees : Single ) : Single;
14342: Function glRadToDeg( const Radians : Extended ) : Extended;
14343: Function glRadToDeg1( const Radians : Single ) : Single;
14344: Function glNormalizeAngle( angle : Single ) : Single
14345: Function glNormalizeDegAngle( angle : Single ) : Single
14346: Procedure glSinCos( const Theta : Extended; var Sin, Cos : Extended );
14347: Procedure glSinCos1( const Theta : Double; var Sin, Cos : Double );
14348: Procedure glSinCos( const Theta : Single; var Sin, Cos : Single );
14349: Procedure glSinCos1( const theta : Double; var Sin, Cos : Extended );
14350: Procedure glSinCos2( const theta, radius : Double; var Sin, Cos : Double );
14351: Procedure glSinCos3( const theta, radius : Single; var Sin, Cos : Single );
14352: Procedure glPrepareSinCosCache( var s, c : array of Single; startAngle, stopAngle : Single )
14353: Function glArcCos( const X : Extended ) : Extended;
14354: Function glArcCos1( const x : Single ) : Single;
14355: Function glArcSin( const X : Extended ) : Extended;
14356: Function glArcSin1( const X : Single ) : Single;
14357: Function glArcTan2l( const Y, X : Extended ) : Extended;
14358: Function glArcTan2l( const Y, X : Single ) : Single;
14359: Function glFastArcTan2( y, x : Single ) : Single
14360: Function glTan( const X : Extended ) : Extended;
14361: Function glTanh( const X : Single ) : Single;
14362: Function glCoTan( const X : Extended ) : Extended;
14363: Function glCoTan1( const X : Single ) : Single;
14364: Function glSinh( const x : Single ) : Single;
14365: Function glSinh1( const x : Double ) : Double;
14366: Function glCosh( const x : Single ) : Single;
14367: Function glCosh1( const x : Double ) : Double;
14368: Function glRSqrt( v : Single ) : Single
14369: Function glRLength( x, y : Single ) : Single
14370: Function glISqrt( i : Integer ) : Integer
14371: Function glILength( x, y : Integer ) : Integer;
14372: Function glILength1( x, y, z : Integer ) : Integer;
14373: Procedure glRegisterBasedExp
14374: Procedure glRandomPointOnSphere( var p : TAffineVector )
14375: Function glRoundInt( v : Single ) : Single;
14376: Function glRoundInt1( v : Extended ) : Extended;
14377: Function glTrunc( v : Single ) : Integer;
14378: Function glTrunc64( v : Extended ) : Int64;
14379: Function glInt( v : Single ) : Single;
14380: Function glInt1( v : Extended ) : Extended;
14381: Function glFrac( v : Single ) : Single;
14382: Function glFract( v : Extended ) : Extended;
14383: Function glRound( v : Single ) : Integer;
14384: Function glRound64( v : Single ) : Int64;
14385: Function glRound641( v : Extended ) : Int64;
14386: Function glTrunc( X : Extended ) : Int64
14387: Function glRound( X : Extended ) : Int64
14388: Function glFrac( X : Extended ) : Extended
14389: Function glCeil( v : Single ) : Integer;
14390: Function glCeil64( v : Extended ) : Int64;
14391: Function glFloor( v : Single ) : Integer;
14392: Function glFloor64( v : Extended ) : Int64;
14393: Function glScaleAndRound( i : Integer; var s : Single ) : Integer
14394: Function glSign( x : Single ) : Integer

```

```

14395: Function glIsInRange( const x, a, b : Single ) : Boolean;
14396: Function glIsInRangeL( const x, a, b : Double ) : Boolean;
14397: Function glIsInCube( const p, d : TAffineVector ) : Boolean;
14398: Function glIsInCubel( const p, d : TVectorGL ) : Boolean;
14399: //Function MinFloat( values : PSingleArray; nbItems : Integer ) : Single;
14400: //Function MinFloat1( values : PDoubleArray; nbItems : Integer ) : Double;
14401: //Function MinFloat2( values : PExtendedArray; nbItems : Integer ) : Extended;
14402: Function glMinFloat3( const v1, v2 : Single ) : Single;
14403: Function glMinFloat4( const v : array of Single ) : Single;
14404: Function glMinFloat5( const v1, v2 : Double ) : Double;
14405: Function glMinFloat6( const v1, v2 : Extended ) : Extended;
14406: Function glMinFloat7( const v1, v2, v3 : Single ) : Single;
14407: Function glMinFloat8( const v1, v2, v3 : Double ) : Double;
14408: Function glMinFloat9( const v1, v2, v3 : Extended ) : Extended;
14409: //Function MaxFloat10( values : PSingleArray; nbItems : Integer ) : Single;
14410: //Function MaxFloat( values : PDoubleArray; nbItems : Integer ) : Double;
14411: //Function MaxFloat1( values : PExtendedArray; nbItems : Integer ) : Extended;
14412: Function glMaxFloat2( const v : array of Single ) : Single;
14413: Function glMaxFloat3( const v1, v2 : Single ) : Single;
14414: Function glMaxFloat4( const v1, v2 : Double ) : Double;
14415: Function glMaxFloat5( const v1, v2 : Extended ) : Extended;
14416: Function glMaxFloat6( const v1, v2, v3 : Single ) : Single;
14417: Function glMaxFloat7( const v1, v2, v3 : Double ) : Double;
14418: Function glMaxFloat8( const v1, v2, v3 : Extended ) : Extended;
14419: Function glMinInteger9( const v1, v2 : Integer ) : Integer;
14420: Function glMinInteger( const v1, v2 : Cardinal ) : Cardinal;
14421: Function glMaxInteger( const v1, v2 : Integer ) : Integer;
14422: Function glMaxInteger1( const v1, v2 : Cardinal ) : Cardinal;
14423: Function glTriangleArea( const p1, p2, p3 : TAffineVector ) : Single;
14424: //Function PolygonArea( const p : PAffineVectorArray; nSides : Integer ) : Single;
14425: Function glTriangleSignedArea( const p1, p2, p3 : TAffineVector ) : Single;
14426: //Function PolygonSignedArea( const p : PAffineVectorArray; nSides : Integer ) : Single;
14427: //Procedure ScaleFloatArray( values : PSingleArray; nb : Integer; var factor : Single );
14428: Procedure glScaleFloatArray( var values : TSingleArray; factor : Single );
14429: //Procedure OffsetFloatArray( values : PSingleArray; nb : Integer; var delta : Single );
14430: Procedure glOffsetFloatArray( var values : array of Single; delta : Single );
14431: //Procedure OffsetFloatArray2( valuesDest, valuesDelta : PSingleArray; nb : Integer );
14432: Function glMaxXYZComponent( const v : TVectorGL ) : Single;
14433: Function glMaxXYZComponent1( const v : TAffineVector ) : single;
14434: Function glMinXYZComponent( const v : TVectorGL ) : Single;
14435: Function glMinXYZComponent1( const v : TAffineVector ) : single;
14436: Function glMaxAbsXYZComponent( v : TVectorGL ) : Single;
14437: Function glMinAbsXYZComponent( v : TVectorGL ) : Single;
14438: Procedure glMaxVector( var v : TVectorGL; const v1 : TVectorGL );
14439: Procedure glMaxVector1( var v : TAffineVector; const v1 : TAffineVector );
14440: Procedure glMinVector( var v : TVectorGL; const v1 : TVectorGL );
14441: Procedure glMinVector1( var v : TAffineVector; const v1 : TAffineVector );
14442: Procedure glSortArrayAscending( var a : array of Extended );
14443: Function glClampValue( const aValue, aMin, aMax : Single ) : Single;
14444: Function glClampValue1( const aValue, aMin : Single ) : Single;
14445: Function glGeometryOptimizationMode : String;
14446: Procedure glBeginFPUOnlySection;
14447: Procedure glEndFPUOnlySection;
14448: Function glConvertRotation( const Angles : TAffineVector ) : TVectorGL;
14449: Function glMakeAffineDblVector( var v : array of Double ) : TAffineDblVector;
14450: Function glMakeDblVector( var v : array of Double ) : THomogeneousDblVector;
14451: Function glVectorAffineDblToFlt( const v : TAffineDblVector ) : TAffineVector;
14452: Function glVectorDblToFlt( const v : THomogeneousDblVector ) : THomogeneousVector;
14453: Function glVectorAffineFltToDbl( const v : TAffineVector ) : TAffineDblVector;
14454: Function glVectorFltToDbl( const v : TVectorGL ) : THomogeneousDblVector;
14455: Function glPointInPolygon( var xp, yp : array of Single; x, y : Single ) : Boolean;
14456: Procedure glDivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word );
14457: Function glTurn( const Matrix : TMatrixGL; angle : Single ) : TMatrixGL;
14458: Function glTurn1( const Matrix : TMatrixGL; const MasterUp:TAffineVector;Angle:Single):TMatrixGL;
14459: Function glPitch( const Matrix : TMatrixGL; Angle : Single ) : TMatrixGL;
14460: Function glPitch1( const Matrix:TMatrixGL;const MasterRight:TAffineVector;Angle:Single):TMatrixGL;
14461: Function glRoll( const Matrix : TMatrixGL; Angle : Single ) : TMatrixGL;
14462: Function glRoll1( const Matrix : TMatrixGL;const MasterDirection:TAffineVector;Angle: Single): TMatrixGL;
14463: Function glRayCastMinDistToPoint( const rayStart, rayVector : TVectorGL;const point:TVectorGL):Single;
14464: Function glRayCastIntersectsSphere( const rayStart, rayVector : TVectorGL; const sphereCenter:TVectorGL;const sphereRadius : Single ) : Boolean;
14465: Function glRayCastSphereIntersect( const rayStart, rayVector : TVectorGL; const sphereCenter:TVectorGL;
14466: const sphereRadius : Single; var i1, i2 : TVectorGL ) : Integer;
14467: Function glSphereVisibleRadius( distance, radius : Single ) : Single;
14468: Function glExtractFrustumFromModelViewProjection( const modelViewProj : TMatrixGL ) : TFrustum;
14469: Function glIsVolumeClipped( const objPos : TVectorGL; const objRadius : Single; const rcci : TRenderContextClippingInfo ) : Boolean;
14470: Function glIsVolumeClipped1( const objPos : TAffineVector; const objRadius : Single; const rcci : TRenderContextClippingInfo ) : Boolean;
14471: Function glIsVolumeClipped2( const min,max:TAffineVector;const rcci : TRenderContextClippingInfo ) : Bool;
14472: Function glMakeParallelProjectionMatrix( const plane : THmgPlane; const dir : TVectorGL ) : TMatrixGL;
14473: Function glMakeShadowMatrix( const planePoint, planeNormal, lightPos : TVectorGL ) : TMatrixGL;
14474: Function glMakeReflectionMatrix( const planePoint, planeNormal : TAffineVector ) : TMatrixGL;
14475: Function glPackRotationMatrix( const mat : TMatrixGL ) : TPackedRotationMatrix;
14476: Function glUnPackRotationMatrix( const packedMatrix : TPackedRotationMatrix ) : TMatrixGL;
14477: 'cPI','Single').setExtended( 3.141592654 );
14478: 'cPIdiv180','Single').setExtended( 0.017453292 );

```

```

14479: 'c180divPI','Single').setExtended( 57.29577951);
14480: 'c2PI','Single').setExtended( 6.283185307);
14481: 'cPIdiv2','Single').setExtended( 1.570796326);
14482: 'cPIdiv4','Single').setExtended( 0.785398163);
14483: 'c3PIdiv4','Single').setExtended( 2.35619449);
14484: 'cInv2PI','Single').setExtended( 1 / 6.283185307);
14485: 'cInv360','Single').setExtended( 1 / 360);
14486: 'c180','Single').setExtended( 180);
14487: 'c360','Single').setExtended( 360);
14488: 'cOneHalf','Single').setExtended( 0.5);
14489: 'cLn10','Single').setExtended( 2.302585093);
14490: {'MinSingle','Extended').setExtended( 1.5e-45);
14491: 'MaxSingle','Extended').setExtended( 3.4e+38);
14492: 'MinDouble','Extended').setExtended( 5.0e-324);
14493: 'MaxDouble','Extended').setExtended( 1.7e+308);
14494: 'MinExtended','Extended').setExtended( 3.4e-4932);
14495: 'MaxExtended','Extended').setExtended( 1.1e+4932);
14496: 'MinComp','Extended').setExtended( - 9.223372036854775807e+18);
14497: 'MaxComp','Extended').setExtended( 9.223372036854775807e+18);}
14498: end;
14499:
14500: procedure SIRegister_GLVectorFileObjects(CL: TPSPPascalCompiler);
14501: begin
14502: AddClassN(FindClass('TOBJECT'), 'TMeshObjectList'
14503: (FindClass('TOBJECT'), 'TFaceGroups'
14504: TMeshAutoCentering', '( macCenterX, macCenterY, macCenterZ, macUseBarycenter )
14505: TMeshAutoCenterings', 'set of TMeshAutoCentering
14506: TMeshObjectMode', '( momTriangles, momTriangleStrip, momFaceGroups )
14507: SIRegister_TBaseMeshObject(CL);
14508: (FindClass('TOBJECT'), 'TSkeletonFrameList'
14509: TSkeletonFrameTransform', '( sftRotation, sftQuaternion )
14510: SIRegister_TSkeletonFrame(CL);
14511: SIRegister_TSkeletonFrameList(CL);
14512: (FindClass('TOBJECT'), 'TSkeleton
14513: (FindClass('TOBJECT'), 'TSkeletonBone
14514: SIRegister_TSkeletonBoneList(CL);
14515: SIRegister_TSkeletonRootBoneList(CL);
14516: SIRegister_TSkeletonBone(CL);
14517: (FindClass('TOBJECT'), 'TSkeletonColliderList
14518: SIRegister_TSkeletonCollider(CL);
14519: SIRegister_TSkeletonColliderList(CL);
14520: (FindClass('TOBJECT'), 'TGLBaseMesh
14521: TBlendedLerpInfo', 'record frameIndex1 : Integer; frameIndex2 : '
14522: +'Integer; lerpFactor : Single; weight : Single; externalPositions : TAffine'
14523: +'VectorList; externalRotations : TAffineVectorList; externalQuaternions : T'
14524: +'QuaternionList; end
14525: SIRegister_TSkeleton(CL);
14526: TMeshObjectRenderingOption', '( moroGroupByMaterial )
14527: TMeshObjectRenderingOptions', 'set of TMeshObjectRenderingOption
14528: SIRegister_TMeshObject(CL);
14529: SIRegister_TMeshObjectList(CL);
14530: //TMeshObjectListClass', 'class of TMeshObjectList
14531: (FindClass('TOBJECT'), 'TMeshMorphTargetList
14532: SIRegister_TMeshMorphTarget(CL);
14533: SIRegister_TMeshMorphTargetList(CL);
14534: SIRegister_TMorphableMeshObject(CL);
14535: TVertexBoneWeight', 'record BoneID : Integer; Weight : Single; end
14536: //PVertexBoneWeightArray', '^TVertexBoneWeightArray // will not work
14537: //PVerticesBoneWeights', '^TVerticesBoneWeights // will not work
14538: TVertexBoneWeightDynArray', 'array of TVertexBoneWeight
14539: SIRegister_TSkeletonMeshObject(CL);
14540: SIRegister_TFaceGroup(CL);
14541: TFaceGroupMeshMode', '( fgmmTriangles, fgmmTriangleStrip, fgmmFl'
14542: +'@Triangles, fgmmTriangleFan, fgmmQuads )
14543: SIRegister_TFGVertexIndexList(CL);
14544: SIRegister_TFGVertexNormalTexIndexList(CL);
14545: SIRegister_TFGIndexTexCoordList(CL);
14546: SIRegister_TFaceGroups(CL);
14547: TMeshNormalsOrientation', '( mnoDefault, mnoInvert )
14548: SIRegister_TVectorFile(CL);
14549: //TVectorFileClass', 'class of TVectorFile
14550: SIRegister_TGLGLSMVectorFile(CL);
14551: SIRegister_TGLBaseMesh(CL);
14552: SIRegister_TGLFreeForm(CL);
14553: TGLActorOption', '( aoSkeletonNormalizeNormals )
14554: TGLActorOptions', 'set of TGLActorOption
14555: 'cDefaultGLActorOptions', 'LongInt').Value.ts32:= ord(aoSkeletonNormalizeNormals);
14556: (FindClass('TOBJECT'), 'TGLActor
14557: TActorAnimationReference', '( aarMorph, aarSkeleton, aarNone )
14558: SIRegister_TActorAnimation(CL);
14559: TActorAnimationName', 'string
14560: SIRegister_TActorAnimations(CL);
14561: SIRegister_TGLBaseAnimationController(CL);
14562: SIRegister_TGLAnimationController(CL);
14563: TActorFrameInterpolation', '( afpNone, afpLinear )
14564: TActorAnimationMode', '( aamNone, aamPlayOnce, aamLoop, aamBounc'
14565: +'eForward, aamBounceBackward, aamLoopBackward, aamExternal )
14566: SIRegister_TGLActor(CL);
14567: SIRegister_TVectorFileFormat(CL);

```

```

14568:  SIRegister_TVectorFileFormatsList(CL);
14569:  (FindClass('TOBJECT'),'EInvalidVectorFile
14570:  Function GetVectorFileFormats : TVectorFileFormatsList
14571:  Function VectorFileFormatsFilter : String
14572:  Function VectorFileFormatsSaveFilter : String
14573:  Function VectorFileFormatExtensionByIndex( index : Integer ) : String
14574:  Procedure RegisterVectorFileFormat( const aExtension, aDescription : String; aClass : TVectorFileClass )
14575:  Procedure UnregisterVectorFileClass( aClass : TVectorFileClass )
14576: end;
14577:
14578: procedure SIRegister_AxCtrls(CL: TPSPascalCompiler);
14579: begin
14580:  'Class_DColorPropPage', 'TGUID '{5CFF5D59-5946-11D0-BDEF-00A024D1875C}
14581:  'Class_DFontPropPage', 'TGUID '{5CFF5D5B-5946-11D0-BDEF-00A024D1875C}
14582:  'Class_DPicturePropPage', 'TGUID '{5CFF5D5A-5946-11D0-BDEF-00A024D1875C}
14583:  'Class_DStringPropPage', 'TGUID '{F42D677E-754B-11D0-BDFB-00A024D1875C}
14584:  SIRegister_TOLEStream(CL);
14585:  (FindClass('TOBJECT'), 'TConnectionPoints
14586:  TConnectionKind', '( ckSingle, ckMulti )
14587:  SIRegister_TConnectionPoint(CL);
14588:  SIRegister_TConnectionPoints(CL);
14589:  TDefinePropertyPage', 'Procedure ( const GUID : TGUID)
14590:  (FindClass('TOBJECT'), 'TActiveXControlFactory
14591:  SIRegister_TActiveXControl(CL);
14592: //TActiveXControlClass', 'class of TActiveXControl
14593:  SIRegister_TActiveXControlFactory(CL);
14594:  SIRegister_TActiveFormControl(CL);
14595:  SIRegister_TActiveForm(CL);
14596: //TActiveFormClass', 'class of TActiveForm
14597:  SIRegister_TActiveFormFactory(CL);
14598:  (FindClass('TOBJECT'), 'TPropertyPageImpl
14599:  SIRegister_TPropertyPage(CL);
14600: //TPropertyPageClass', 'class of TPropertyPage
14601:  SIRegister_TPropertyPageImpl(CL);
14602:  SIRegister_TActiveXPropertyPage(CL);
14603:  SIRegister_TActiveXPropertyPageFactory(CL);
14604:  SIRegister_TCustomAdapter(CL);
14605:  SIRegister_TAdapterNotifier(CL);
14606:  SIRegister_TFontAccess(CL);
14607:  SIRegister_TFontAdapter(CL);
14608:  SIRegister_TPictureAccess(CL);
14609:  SIRegister_TPictureAdapter(CL);
14610:  SIRegister_TOleGraphic(CL);
14611:  SIRegister_TStringsAdapter(CL);
14612:  SIRegister_TReflectorWindow(CL);
14613:  Procedure EnumDispatchProperties(Dispatch:IDispatch;PropType:GUID;VTCode:Int;PropList:TStrings);
14614:  Procedure GetOleFont( Font : TFont; var OleFont : IFontDisp)
14615:  Procedure SetOleFont( Font : TFont; OleFont : IFontDisp)
14616:  Procedure GetOlePicture( Picture : TPicture; var OlePicture : IPictureDisp)
14617:  Procedure SetOlePicture( Picture : TPicture; Olepicture : IPictureDisp)
14618:  Procedure GetOleStrings( Strings : TStrings; var OleStrings : IStrings)
14619:  Procedure SetOleStrings( Strings : TStrings; OleStrings : IStrings)
14620:  Function ParkingWindow : HWND
14621: end;
14622:
14623: procedure SIRegister_synaip(CL: TPSPascalCompiler);
14624: begin
14625: // TIP6Bytes = array [0..15] of Byte;
14626: {:binary form of IPv6 adress (for string conversion routines)}
14627: // TIP6Words = array [0..7] of Word;
14628: AddTypeS('TIP6Bytes', 'array [0..15] of Byte;');
14629: AddTypeS('TIP6Words', 'array [0..7] of Word;');
14630: AddTypeS('TIPAdr', 'record Oct1 : Byte; Oct2 : Byte; Oct3 : Byte; Oct4: Byte; end');
14631: Function synaIsIP( const Value : string ) : Boolean';
14632: Function synaIsIP6( const Value : string ) : Boolean';
14633: Function synaIPToID( Host : string ) : Ansistring';
14634: Function synaStrToIp6( value : string ) : TIP6Bytes';
14635: Function synaIp6ToStr( value : TIP6Bytes ) : string';
14636: Function synaStrToIp( value : string ) : integer';
14637: Function synaIpToStr( value : integer ) : string';
14638: Function synaReverseIP( Value : AnsiString ) : AnsiString';
14639: Function synaReverseIP6( Value : AnsiString ) : AnsiString';
14640: Function synaExpandIP6( Value : AnsiString ) : AnsiString';
14641: Function xStrToIP( const Value : String ) : TIPAdr';
14642: Function xIPToStr( const Adresse : TIPAdr ) : String';
14643: Function IPTOCardinal( const Adresse : TIPAdr ) : Cardinal';
14644: Function CardinalToIP( const Value : Cardinal ) : TIPAdr';
14645: end;
14646:
14647: procedure SIRegister_synacode(CL: TPSPascalCompiler);
14648: begin
14649: AddTypeS('TSpecials', 'set of Char');
14650: Const ('SpecialChar','TSpecials').SetSet( '={[]}<>;:@/?\\"_');
14651: Const ('URLFullSpecialChar','TSpecials').SetSet( ':/?:@=&#+');
14652: Const ('TableBasee64' ABCDEFGHIJKLMNOPQRSTUVWXYZabcdeffghi jklmnopqrstuvwxyz0123456789+/=+');
14653: Const ('TableBasee64mod' ABCDEFGHIJKLMNOPQRSTUVWXYZabcdeffghi jklmnopqrstuvwxyz0123456789+,=+');
14654: Const ('TableUU' ('!#$%&!' ()*+,-./0123456789:;<=>@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_');
14655: Const ('TableXX' (+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdeffghi jklmnopqrstuvwxyz');
14656: Function DecodeTriplet( const Value : AnsiString; Delimiter : Char ) : AnsiString');
```

```

14657: Function DecodeQuotedPrintable( const Value : AnsiString ) : AnsiString';
14658: Function DecodeURL( const Value : AnsiString ) : AnsiString';
14659: Function EncodeTriplet( const Value : AnsiString; Delimiter : Char; Specials : TSpecials ) : AnsiString';
14660: Function EncodeQuotedPrintable( const Value : AnsiString ) : AnsiString';
14661: Function EncodeSafeQuotedPrintable( const Value : AnsiString ) : AnsiString';
14662: Function EncodeURLElement( const Value : AnsiString ) : AnsiString';
14663: Function EncodeURL( const Value : AnsiString ) : AnsiString';
14664: Function Decode4to3( const Value, Table : AnsiString ) : AnsiString';
14665: Function Decode4to3Ex( const Value, Table : AnsiString ) : AnsiString';
14666: Function Encode3to4( const Value, Table : AnsiString ) : AnsiString';
14667: Function synDecodeBase64( const Value : AnsiString ) : AnsiString';
14668: Function synEncodeBase64( const Value : AnsiString ) : AnsiString';
14669: Function DecodeBase64mod( const Value : AnsiString ) : AnsiString );
14670: Function EncodeBase64mod( const Value : AnsiString ) : AnsiString );
14671: Function DecodeUU( const Value : AnsiString ) : AnsiString');
14672: Function EncodeUU( const Value : AnsiString ) : AnsiString');
14673: Function DecodeXX( const Value : AnsiString ) : AnsiString');
14674: Function DecodeYEnc( const Value : AnsiString ) : AnsiString');
14675: Function UpdateCrc32( Value : Byte; Crc32 : Integer ) : Integer');
14676: Function synCrc32( const Value : AnsiString ) : Integer');
14677: Function UpdateCrc16( Value : Byte; Crc16 : Word ) : Word');
14678: Function Crc16( const Value : AnsiString ) : Word');
14679: Function synMD5( const Value : AnsiString ) : Ansistring');
14680: Function HMAC_MD5( Text, Key : AnsiString ) : AnsiString');
14681: Function MD5LongHash( const Value : AnsiString; Len : integer ) : AnsiString');
14682: Function synSHA1( const Value : AnsiString ) : AnsiString');
14683: Function HMAC_SHA1( Text, Key : AnsiString ) : AnsiString');
14684: Function SHA1LongHash( const Value : AnsiString; Len : integer ) : AnsiString');
14685: Function synMD4( const Value : AnsiString ) : Ansistring');
14686: end;
14687:
14688: procedure SIRegister_synachar(CL: TPSPascalCompiler);
14689: begin
14690:   AddTypeS('TMimeChar', '( ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, I'
14691:             + 'SO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, IS'
14692:             + 'O_8859_13, ISO_8859_14, ISO_8859_15, CP1250, CP1251, CP1252, CP1253, CP125'
14693:             + '4, CP1255, CP1256, CP1257, CP1258, KOI8_R, CP895, CP852, UCS_2, UCS_4, UTF'
14694:             + '_8, UTF_7, UTF_7mod, UCS_2LE, UCS_4LE, UTF_16, UTF_16LE, UTF_32, UTF_32LE, '
14695:             + 'C99, JAVA, ISO_8859_16, KOI8_U, KOI8_RU, CP862, CP866, MAC, MACCE, MACICE'
14696:             + ', MACCRO, MACRO, MACCYR, MACUK, MACGR, MACTU, MACHEB, MACAR, MACTH, ROMAN8'
14697:             + ', NEXTSTEP, ARMASCII, GEORGIAN_AC, GEORGIAN_PS, KOI8_T, MULELAO, CP1133, T'
14698:             + 'IS620, CP874, VISCII, TCVN, ISO_IR_14, JIS_X0201, JIS_X0208, JIS_X0212, GB'
14699:             + '1988_80, GB2312_80, ISO_IR_165, ISO_IR_149, EUC_JP, SHIFT_JIS, CP932, ISO_'
14700:             + '2022_JP, ISO_2022_JP1, ISO_2022_JP2, GB2312, CP936, GB18030, ISO_2022_CN, '
14701:             + 'ISO_2022_CNE, HZ, EUC_TW, BIG5, CP950, BIG5_HKSCS, EUC_KR, CP949, CP1361, '
14702:             + 'ISO_2022_KR, CP737, CP775, CP853, CP855, CP857, CP858, CP860, CP861, CP863'
14703:             + ', CP864, CP865, CP869, CP1125 ') );
14704:   AddTypeS('TMimeSetChar', 'set of TMimeChar');
14705:   Function CharsetConversion(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar):AnsiString;
14706:   Function CharsetConversionEx(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeChar; const
14707:     TransformTable : array of Word) : AnsiString');
14708:   Function CharsetConversionTrans( Value : AnsiString; CharFrom : TMimeChar; CharTo : TMimeChar; const
14709:     TransformTable : array of Word; Translit : Boolean) : AnsiString');
14710:   Function GetCurCP : TMimeChar');
14711:   Function GetCurOEMCP : TMimeChar');
14712:   Function GetCPFromID( Value : AnsiString ) : TMimeChar');
14713:   Function GetIDFromCP( Value : TMimeChar ) : AnsiString');
14714:   Function NeedCharsetConversion( const Value : AnsiString ) : Boolean');
14715:   Function IdealCharsetCoding(const Value:AnsiString;CharFrom:TMimeChar;CharTo:TMimeSetChar):TMimeChar;
14716:   Function GetBOM( Value : TMimeChar ) : AnsiString');
14717:   Function StringToWide( const Value : AnsiString ) : WideString');
14718:   Function WideToString( const Value : WideString ) : AnsiString');
14719: end;
14720:
14721: procedure SIRegister_synamisc(CL: TPSPascalCompiler);
14722: begin
14723:   AddTypeS('TProxySetting', 'record Host : string; Port : string; Bypass : string; end');
14724:   Procedure WakeOnLan( MAC, IP : string );
14725:   Function GetDNS : string');
14726:   Function GetIEProxy( protocol : string ) : TProxySetting');
14727:   Function GetLocalIPs : string');
14728:
14729: procedure SIRegister_synaser(CL: TPSPascalCompiler);
14730: begin
14731:   AddConstantN('synCR', 'Char #$0d);
14732:   Const('synLF', 'Char #$0a);
14733:   Const('cSerialChunk', 'LongInt'( 8192));
14734:   Const('LockfileDirectory', 'String '/var/lock');
14735:   Const('PortIsClosed', 'LongInt'(- 1));
14736:   Const('ErrAlreadyOwned', 'LongInt'( 9991);
14737:   Const('ErrAlreadyInUse', 'LongInt'( 9992);
14738:   Const('ErrWrongParameter', 'LongInt'( 9993);
14739:   Const('ErrPortNotOpen', 'LongInt'( 9994);
14740:   Const('ErrNoDeviceAnswer', 'LongInt'( 9995);
14741:   Const('ErrMaxBuffer', 'LongInt'( 9996);
14742:   Const('ErrTimeout', 'LongInt'( 9997);
14743:   Const('ErrNotRead', 'LongInt'( 9998);

```

```

14744: Const('ErrFrame','LongInt'( 9999);
14745: Const('ErrOverrun','LongInt'( 10000);
14746: Const('ErrRxOver','LongInt'( 10001);
14747: Const('ErrRxParity','LongInt'( 10002);
14748: Const('ErrTxFull','LongInt'( 10003);
14749: Const('dcb_Binary','LongWord')( $00000001);
14750: Const('dcb_ParityCheck','LongWord')( $00000002);
14751: Const('dcb_OutxCtsFlow','LongWord')( $00000004);
14752: Const('dcb_OutxDsrFlow','LongWord')( $00000008);
14753: Const('dcb_DtrControlMask','LongWord')( $00000030);
14754: Const('dcb_DtrControlDisable','LongWord')( $00000000);
14755: Const('dcb_DtrControlEnable','LongWord')( $00000010);
14756: Const('dcb_DtrControlHandshake','LongWord')( $00000020);
14757: Const('dcb_DsrSensitivity','LongWord')( $00000040);
14758: Const('dcb_TXContinueOnXoff','LongWord')( $00000080);
14759: Const('dcb_OutX','LongWord')( $00000100);
14760: Const('dcb_InX','LongWord')( $00000200);
14761: Const('dcb_ErrorChar','LongWord')( $00000400);
14762: Const('dcb_NullStrip','LongWord')( $00000800);
14763: Const('dcb_RtsControlMask','LongWord')( $00003000);
14764: Const('dcb_RtsControlDisable','LongWord')( $00000000);
14765: Const('dcb_RtsControlEnable','LongWord')( $00001000);
14766: Const('dcb_RtsControlHandshake','LongWord')( $00002000);
14767: Const('dcb_RtsControlToggle','LongWord')( $00003000);
14768: Const('dcb_AbortOnError','LongWord')( $00004000);
14769: Const('dcb_Reserveds','LongWord')( $FFFF8000);
14770: Const('synSBL','LongInt'( 0);
14771: Const('SB1andHalf','LongInt'( 1);
14772: Const('synSB2','LongInt'( 2);
14773: Const('synINVALID_HANDLE_VALUE','LongInt'( THandle ( - 1 )));
14774: Const('CS7fix','LongWord')( $0000020);
14775: AddTypeS('synTDCB', 'record DCBlength : DWORD; BaudRate : DWORD; Flags : Long'
14776: +'int; wReserved : Word; XonLim : Word; XoffLim : Word; ByteSize : Byte; Par'
14777: +'ity : Byte; StopBits : Byte; XonChar : CHAR; XoffChar : CHAR; ErrorChar : '
14778: +'CHAR; EofChar : CHAR; EvtChar : CHAR; wReserved1 : Word; end');
14779: //AddTypeS('PDCB', '^TDCB // will not work');
14780: //Const('MaxRates','LongInt'( 18);
14781: //Const('MaxRates','LongInt'( 30);
14782: //Const('MaxRates','LongInt'( 19);
14783: Const('O_SYNC','LongWord')( $0080);
14784: Const('synOK','LongInt'( 0);
14785: Const('synErr','LongInt'( integer ( - 1 )));
14786: AddTypeS('THookSerialReason', '( HR_SerialClose, HR_Connect, HR_CanRead, HR_CanWrite, HR_ReadCount,
14787: HR_WriteCount, HR_Wait )');
14788: Type('THookSerialStatus',Procedure(Sender: TObject; Reason:THookSerialReason; const Value:string));
14789: SIRegister_ESynaSerError(CL);
14790: SIRegister_TBlockSerial(CL);
14791: Function GetSerialPortNames : string;
14792: end;
14793: procedure SIRegister_synaicnv(CL: TPSCompiler);
14794: begin
14795: Const('DLLIconvName','String 'libiconv.so');
14796: Const('DLLIconvName','String 'iconv.dll');
14797: AddTypeS('size_t','Cardinal');
14798: AddTypes('iconv_t','Integer');
14799: //AddTypeS('iconv_t','Pointer');
14800: AddTypeS('argptr','iconv_t');
14801: Function SynalIconvOpen( const tocode, fromcode : Ansistring) : iconv_t';
14802: Function SynalIconvOpenTranslit( const tocode, fromcode : Ansistring) : iconv_t';
14803: Function SynalIconvOpenIgnore( const tocode, fromcode : Ansistring) : iconv_t';
14804: Function SynalIconv( cd : iconv_t; inbuf : AnsiString; var outbuf : AnsiString) : integer';
14805: Function SynalIconvClose( var cd : iconv_t) : integer';
14806: Function SynalIconvCtl( cd : iconv_t; request : integer; argument : argptr) : integer';
14807: Function IsIconvloaded : Boolean';
14808: Function InitIconvInterface : Boolean';
14809: Function DestroyIconvInterface : Boolean';
14810: Const('ICONV_TRIVIALP','LongInt'( 0);
14811: Const('ICONV_GET_TRANSLITERATE','LongInt'( 1);
14812: Const('ICONV_SET_TRANSLITERATE','LongInt'( 2);
14813: Const('ICONV_GET_DISCARD_ILSEQ','LongInt'( 3);
14814: Const('ICONV_SET_DISCARD_ILSEQ','LongInt'( 4);
14815: end;
14816:
14817: procedure SIRegister_pingsend(CL: TPSCompiler);
14818: begin
14819: Const 'ICMP_ECHO','LongInt'( 8);
14820: Const('ICMP_ECHOREPLY','LongInt'( 0);
14821: Const('ICMP_UNREACH','LongInt'( 3);
14822: Const('ICMP_TIME_EXCEEDED','LongInt'( 11);
14823: Const('ICMP6_ECHO','LongInt'( 128);
14824: Const('ICMP6_ECHOREPLY','LongInt'( 129);
14825: Const('ICMP6_UNREACH','LongInt'( 1);
14826: Const('ICMP6_TIME_EXCEEDED','LongInt'( 3);
14827: AddTypeS('TICMPError', '( IE_NoError, IE_Other, IE_TTLExceed, IE_UnreachOut'
14828: +'her, IE_UnreachRoute, IE_UnreachAdmin, IE_UnreachAddr, IE_UnreachPort )');
14829: SIRegister_TPINGSend(CL);
14830: Function PingHost( const Host : string) : Integer';
14831: Function TraceRouteHost( const Host : string) : string';

```

```

14832: end;
14833:
14834: procedure SIRegister_asnlutil(CL: TPSPascalCompiler);
14835: begin
14836:   AddConstantN('synASN1_BOOL','LongWord')( $01);
14837:   Const('synASN1_INT','LongWord')( $02);
14838:   Const('synASN1_OCTSTR','LongWord')( $04);
14839:   Const('synASN1_NULL','LongWord')( $05);
14840:   Const('synASN1_OBJID','LongWord')( $06);
14841:   Const('synASN1_ENUM','LongWord')( $0a);
14842:   Const('synASN1_SEQ','LongWord')( $30);
14843:   Const('synASN1_SETOF','LongWord')( $31);
14844:   Const('synASN1_IPADDR','LongWord')( $40);
14845:   Const('synASN1_COUNTER','LongWord')( $41);
14846:   Const('synASN1_GAUGE','LongWord')( $42);
14847:   Const('synASN1_TIMETICKS','LongWord')( $43);
14848:   Const('synASN1_OPAQUE','LongWord')( $44);
14849:   Function synASNEncOIDItem( Value : Integer ) : AnsiString');
14850:   Function synASNDecOIDItem( var Start : Integer; const Buffer : AnsiString ) : Integer');
14851:   Function synASNEncLen( Len : Integer ) : AnsiString');
14852:   Function synASNDecLen( var Start : Integer; const Buffer : AnsiString ) : Integer');
14853:   Function synASNEncInt( Value : Integer ) : AnsiString');
14854:   Function synASNEncUInt( Value : Integer ) : AnsiString');
14855:   Function synASNObject( const Data: AnsiString; ASNTYPE : Integer ) : AnsiString');
14856:   Function synASNItem(var Start:Integer;const Buffer:AnsiString;var ValueType:Integer):AnsiString;
14857:   Function synMibToId( Mib : String ) : AnsiString');
14858:   Function synIdToMib( const Id : AnsiString ) : String');
14859:   Function synIntMibToStr( const Value : AnsiString ) : AnsiString');
14860:   Function ASNdump( const Value : AnsiString ) : AnsiString');
14861:   Function GetMailServers( const DNSHost, Domain : AnsiString; const Servers:TStrings) : Boolean');
14862:   Function LDAPResultDump( const Value : TLDAPResultList ) : AnsiString');
14863: end;
14864:
14865: procedure SIRegister_ldapsend(CL: TPSPascalCompiler);
14866: begin
14867:   Const('cLDAPProtocol','String '389');
14868:   Const('LDAP ASN1 BIND REQUEST','LongWord')( $60);
14869:   Const('LDAP ASN1 BIND RESPONSE','LongWord')( $61);
14870:   Const('LDAP ASN1 UNBIND REQUEST','LongWord')( $42);
14871:   Const('LDAP ASN1 SEARCH REQUEST','LongWord')( $63);
14872:   Const('LDAP ASN1 SEARCH ENTRY','LongWord')( $64);
14873:   Const('LDAP ASN1 SEARCH DONE','LongWord')( $65);
14874:   Const('LDAP ASN1 SEARCH REFERENCE','LongWord')( $73);
14875:   Const('LDAP ASN1 MODIFY REQUEST','LongWord')( $66);
14876:   Const('LDAP ASN1 MODIFY RESPONSE','LongWord')( $67);
14877:   Const('LDAP ASN1 ADD REQUEST','LongWord')( $68);
14878:   Const('LDAP ASN1 ADD RESPONSE','LongWord')( $69);
14879:   Const('LDAP ASN1 DEL REQUEST','LongWord')( $4A);
14880:   Const('LDAP ASN1 DEL RESPONSE','LongWord')( $6B);
14881:   Const('LDAP ASN1 MODIFYDN REQUEST','LongWord')( $6C);
14882:   Const('LDAP ASN1 MODIFYDN RESPONSE','LongWord')( $6D);
14883:   Const('LDAP ASN1 COMPARE REQUEST','LongWord')( $6E);
14884:   Const('LDAP ASN1 COMPARE RESPONSE','LongWord')( $6F);
14885:   Const('LDAP ASN1 ABANDON REQUEST','LongWord')( $70);
14886:   Const('LDAP ASN1 EXT REQUEST','LongWord')( $77);
14887:   Const('LDAP ASN1 EXT RESPONSE','LongWord')( $78);
14888:   SIRegister_TLDAPAttribute(CL);
14889:   SIRegister_TLDAPAttributeList(CL);
14890:   SIRegister_TLDAPResult(CL);
14891:   SIRegister_TLDAPResultList(CL);
14892:   AddTypeS('TLDAPModifyOp', '( MO_Add, MO_Delete, MO_Replace )');
14893:   AddTypeS('TLDAPSearchScope', '( SS_BaseObject, SS_SingleLevel, SS_WholeSubtree )');
14894:   AddTypeS('TLDAPSearchAliases', '( SA_NeverDeref, SA_InSearching, SA_FindingBaseObj, SA_Always )');
14895:   SIRegister_TLDAPSnd(CL);
14896:   Function LDAPResultDump( const Value : TLDAPResultList ) : AnsiString');
14897: end;
14898:
14899:
14900: procedure SIRegister_slogsend(CL: TPSPascalCompiler);
14901: begin
14902:   Const('cSysLogProtocol','String '514');
14903:   Const('FCL_Kernel','LongInt'( 0);
14904:   Const('FCL_UserLevel','LongInt'( 1);
14905:   Const('FCL_MailSystem','LongInt'( 2);
14906:   Const('FCL_System','LongInt'( 3);
14907:   Const('FCL_Security','LongInt'( 4);
14908:   Const('FCL_Syslogd','LongInt'( 5);
14909:   Const('FCL_Printer','LongInt'( 6);
14910:   Const('FCL_News','LongInt'( 7);
14911:   Const('FCL_UUCP','LongInt'( 8);
14912:   Const('FCL_Clock','LongInt'( 9);
14913:   Const('FCL_Authorization','LongInt'( 10);
14914:   Const('FCL_FTP','LongInt'( 11);
14915:   Const('FCL_NTP','LongInt'( 12);
14916:   Const('FCL_LogAudit','LongInt'( 13);
14917:   Const('FCL_LogAlert','LongInt'( 14);
14918:   Const('FCL_Time','LongInt'( 15);
14919:   Const('FCL_Local0','LongInt'( 16);
14920:   Const('FCL_Locall1','LongInt'( 17);

```

```

14921: Const ('FCL_Local2','LongInt'( 18);
14922: Const ('FCL_Local3','LongInt'( 19);
14923: Const ('FCL_Local4','LongInt'( 20);
14924: Const ('FCL_Local5','LongInt'( 21);
14925: Const ('FCL_Local6','LongInt'( 22);
14926: Const ('FCL_Local7','LongInt'( 23);
14927: Type(TSyslogSeverity,'(Emergency, Alert, Critical, Error, Warning,Notice,Info,Debug);
14928: SIRegister_TSyslogMessage(CL);
14929: SIRegister_TSyslogSend(CL);
14930: Function ToSysLog(const SyslogServer:string;Facil:Byte;Sever:TSyslogSeverity;const
Content:string):Boolean;
14931: end;
14932:
14933:
14934: procedure SIRegister_mimemess(CL: TPSPascalCompiler);
14935: begin
14936:   AddTypeS('TMessPriority', '( MP_unknown, MP_low, MP_normal, MP_high )');
14937:   SIRegister_TMessHeader(CL);
14938:   //AddTypeS('TMessHeaderClass', 'class of TMessHeader');
14939:   SIRegister_TMimeMess(CL);
14940: end;
14941:
14942: procedure SIRegister_mimepart(CL: TPSPascalCompiler);
14943: begin
14944:   (FindClass('TOBJECT'), 'TMimePart');
14945:   AddTypeS('THookWalkPart', 'Procedure ( const Sender : TMimePart)');
14946:   AddTypeS('TMimePrimary', '( MP_TEXT, MP_MULTIPART, MP_MESSAGE, MP_BINARY )');
14947:   AddTypeS('TMimeEncoding', '( ME_7BIT, ME_8BIT, ME_QUOTED_PRINTABLE, ME_BASE64, ME_UU, ME_XX )');
14948:   SIRegister_TMimePart(CL);
14949:   Const ('MaxMimeType','LongInt'( 25);
14950:   Function GenerateBoundary : string');
14951: end;
14952:
14953: procedure SIRegister_mimeinln(CL: TPSPascalCompiler);
14954: begin
14955:   Function InlineDecode( const Value : string; CP : TMimeChar ) : string';
14956:   Function InlineEncode( const Value : string; CP, MimeP : TMimeChar ) : string');
14957:   Function NeedInline( const Value : AnsiString ) : boolean');
14958:   Function InlineCodeEx( const Value : string; FromCP : TMimeChar ) : string');
14959:   Function InlineCode( const Value : string ) : string');
14960:   Function InlineEmailEx( const Value : string; FromCP : TMimeChar ) : string');
14961:   Function InlineEmail( const Value : string ) : string');
14962: end;
14963:
14964: procedure SIRegister_ftpsend(CL: TPSPascalCompiler);
14965: begin
14966:   Const ('cFtpProtocol','String '21');
14967:   Const ('cFtpDataProtocol','String '20');
14968:   Const ('FTP_OK','LongInt'( 255);
14969:   Const ('FTP_ERR','LongInt'( 254);
14970:   AddTypeS('TFTPStatus', 'Procedure ( Sender : TObject; Response : Boolean; const Value : string)');
14971:   SIRegister_TFTPListRec(CL);
14972:   SIRegister_TFTPList(CL);
14973:   SIRegister_TFTPSend(CL);
14974:   Function FtpGetFile( const IP, Port, FileName, LocalFile, User, Pass : string ) : Boolean';
14975:   Function FtpPutFile( const IP, Port, FileName, LocalFile, User, Pass : string ) : Boolean';
14976:   Function FtpInterServerTransfer(const FromIP,FromPort,FromFile,FromUser,FromPass : string; const ToIP,
ToPort,ToFile,ToUser,ToPass : string ) : Boolean';
14977: end;
14978:
14979: procedure SIRegister_httpsend(CL: TPSPascalCompiler);
14980: begin
14981:   Const ('cHttpProtocol','String '80');
14982:   AddTypeS('TTransferEncoding', '( TE_UNKNOWN, TE_IDENTITY, TE_CHUNKED )');
14983:   SIRegister_THTTPSend(CL);
14984:   Function HttpGetText( const URL : string; const Response : TStrings ) : Boolean';
14985:   Function HttpGetBinary( const URL : string; const Response : TStream ) : Boolean';
14986:   Function HttpPostBinary( const URL : string; const Data : TStream ) : Boolean');
14987:   Function HttpPostURL( const URL, URLData : string; const Data : TStream ) : Boolean');
14988:   Function HttpPostfile(const URL,FieldName,FileName:string;const Data:TStream;const
ResultData:TStrings):Bool;
14989: end;
14990:
14991: procedure SIRegister_smtpsend(CL: TPSPascalCompiler);
14992: begin
14993:   Const ('cSmtpProtocol','String '25');
14994:   SIRegister_TSMTSPSend(CL);
14995:   Function SendToRaw(const MailFr,MailTo,SMTPHost:string;const MailData:TStrings;const Usrname,
Passw:string):Bool;
14996:   Function SendTo(const MailFrom,MailTo,Subject,SMTPHost:string;const MailData:TStrings):Boolean;
14997:   Function SendToEx(const MailFrom, MailTo, Subject, SMTPHost : string; const MailData : TStrings; const
Usrname, Password : string):Boolean';
14998: end;
14999:
15000: procedure SIRegister_snmpsend(CL: TPSPascalCompiler);
15001: begin
15002:   Const ('cSnmpProtocol','String '161');
15003:   Const ('cSnmpTrapProtocol','String '162');
15004:   Const ('SNMP_V1','LongInt'( 0 );

```

```

15005: Const('SNMP_V2C','LongInt'( 1);
15006: Const('SNMP_V3','LongInt'( 3);
15007: Const('PDUGetRequest','LongWord')( $A0);
15008: Const('PDUGetNextRequest','LongWord')( $A1);
15009: Const('PDUGetResponse','LongWord')( $A2);
15010: Const('PDUSetRequest','LongWord')( $A3);
15011: Const('PDUTrap','LongWord')( $A4);
15012: Const('PDUGetBulkRequest','LongWord')( $A5);
15013: Const('PDUInformRequest','LongWord')( $A6);
15014: Const('PDUTrapV2','LongWord')( $A7);
15015: Const('PDUReport','LongWord')( $A8);
15016: Const('ENOError',LongInt 0;
15017: Const('ETooBig','LongInt')( 1);
15018: Const('ENoSuchName','LongInt'( 2);
15019: Const('EBadValue','LongInt'( 3);
15020: Const('EReadOnly','LongInt'( 4);
15021: Const('EGenErr','LongInt'( 5);
15022: Const('ENoAccess','LongInt'( 6);
15023: Const('EWrongType','LongInt'( 7);
15024: Const('EWrongLength','LongInt'( 8);
15025: Const('EWrongEncoding','LongInt'( 9);
15026: Const('EWrongValue','LongInt'( 10);
15027: Const('ENOCreation','LongInt'( 11);
15028: Const('EInconsistentValue','LongInt'( 12);
15029: Const('EResourceUnavailable','LongInt'( 13);
15030: Const('ECommitFailed','LongInt'( 14);
15031: Const('EUndoFailed','LongInt'( 15);
15032: Const('EAutorizationError','LongInt'( 16);
15033: Const('ENotWritable','LongInt'( 17);
15034: Const('EInconsistentName','LongInt'( 18);
15035: AddTypes('TV3Flags', '( NoAuthNoPriv, AuthNoPriv, AuthPriv )');
15036: AddTypes('TV3Auth', '( AuthMD5, AuthSHA1 )');
15037: AddTypes('TV3Priv', '( PrivDES, Priv3DES, PrivAES )');
15038: SIRegister_TSNSMPMib(CL);
15039: AddTypes('TV3Sync', 'record EngineID : AnsiString; EngineBoots : integer;
15040:   +'EngineTime : integer; EngineStamp : Cardinal; end');
15041: SIRegister_TSNSMPRec(CL);
15042: SIRegister_TSNSMPSend(CL);
15043: Function SNMPGet( const OID, Community, SNMPHost : AnsiString; var Value : AnsiString) : Boolean';
15044: Function SNMPSet( const OID, Community, SNMPHost, Value : AnsiString; ValueType : Integer) : Boolean';
15045: Function SNMPGetNext(var OID:AnsiString;const Community,SNMPHost:AnsiString;var Value:AnsiString):Boolean;
15046: Function SNMPGetTable(const BaseOID, Community, SNMPHost : AnsiString; const Value : TStrings): Boolean;
15047: Function SNMPGetTableElement(const BaseOID,RowID,Colid,Community,SNMPHost:AnsiStr;var Value:AnsiStr):Bool;
15048: Function SendTrap(const Dest,Source,Enterprise,Community : AnsiString; Generic, Specific, Seconds : Integer;
15049:   const MIBName, MIBValue : AnsiString; MIBType : Integer);
15050: end;
15051:
15052: procedure SIRegister_NetWork(CL: TPSPascalCompiler);
15053: begin
15054:   Function GetDomainName2: AnsiString');
15055:   Function GetDomainController( Domain : AnsiString) : AnsiString');
15056:   Function GetDomainUsers( Controller : AnsiString) : AnsiString');
15057:   Function GetDomainGroups( Controller : AnsiString) : AnsiString');
15058:   Function GetDateTime( Controller : AnsiString) : TDateTime');
15059:   Function GetFullName2( Controller, UserID : AnsiString) : AnsiString');
15060: end;
15061:
15062: procedure SIRegister_wwSystem(CL: TPSPascalCompiler);
15063: begin
15064:   AddTypes('TwwDateOrder', '( doMDY, doDMY, doYMD )');
15065:   TwwDateTimeSelection', '(wwdsDay,wwdsMonth,wwdsYear,wwdsHour,wwdsMinute,wwdsSecond,wwdsAMPM);
15066:   Function wwStrToDate( const S : string) : boolean');
15067:   Function wwStrToTime( const S : string) : boolean');
15068:   Function wwStrToDateTime( const S : string) : boolean');
15069:   Function wwStrToDateVal( const S : string) : TDateTime');
15070:   Function wwStrToDateVal( const S : string) : TDateTime');
15071:   Function wwStrToDateTimeVal( const S : string) : TDateTime');
15072:   Function wwStrToInt( const S : string) : boolean');
15073:   Function wwStrToFloat( const S : string) : boolean');
15074:   Function wwGetDateOrder( const DateFormat : string) : TwwDateOrder');
15075:   Function wwNextDay( Year, Month, Day : Word) : integer');
15076:   Function wwPriorDay( Year, Month, Day : Word) : integer');
15077:   Function wwDoEncodeDate( Year, Month, Day : Word; var Date : TDateTime) : Boolean');
15078:   Function wwDoEncodeTime( Hour, Min, Sec, MSec : Word; var Time : TDateTime) : Boolean');
15079:   Function wwGetDateTimeCursorPosition(SelStart:int;Text:string;TimeOnly:Bool):TwwDateTimeSelection');
15080:   Function wwGetTimeCursorPosition( SelStart : integer; Text : string) : TwwDateTimeSelection');
15081:   Function wwScanDate( const S : string; var Date : TDateTime) : Boolean');
15082:   Function wwScanDateEpoch( const S : string; var Date : TDateTime; Epoch : integer) : Boolean');
15083:   Procedure wwSetDateTimeCursorPosition(dateCursor:TwwDateTimeSelection;edit:TCustomEdit;TimeOnly:Bool);
15084:   Function wwStrToFloat2( const S : string; var FloatValue : Extended; DisplayFormat: string): boolean');
15085: end;
15086:
15087: unit uPSI_Themes;
15088: Function ThemeServices : TThemeServices');
15089: Function ThemeControl( AControl : TControl) : Boolean');
15090: Function UnthemedDesigner( AControl : TControl) : Boolean');
15091: procedure SIRegister_UDDIHelper(CL: TPSPascalCompiler);

```

```

15092: begin
15093:   Function GetBindingkeyAccessPoint(const Operator : String; const key : String) : String');
15094: end;
15095: Unit uPSC_menus;
15096:   Function StripHotkey( const Text : string ) : string');
15097:   Function GetHotkey( const Text : string ) : string');
15098:   Function AnsiSameCaption( const Text1, Text2 : string ) : Boolean');
15099:   Function IsAltGRPressed : boolean');
15100:
15101: procedure SIRегистer_IdIMAP4Server(CL: TPSPPascalCompiler);
15102: begin
15103:   TCommandEvent', 'Procedure(Thread:TIdPeerThread; const Tag,CmdStr:String;var Handled:Boolean);
15104:   SIRегистer_TIdIMAP4Server(CL);
15105: end;
15106:
15107: procedure SIRегистer_VariantSymbolTable(CL: TPSPPascalCompiler);
15108: begin
15109:   'HASH_SIZE', 'LongInt'( 256 );
15110:   CL.FindClass('TOBJECT','EVariantSymbolTable');
15111:   CL.AddTypeS('TSymbolType', '( stInteger, stFloat, stDate, stString )');
15112:   //CL.AddTypeS('PSymbol', '^TSymbol // will not work');
15113:   CL.AddTypeS('TSymbol', 'record Name : String; BlockLevel : integer; HashValue'
15114:     +' : Integer; Value : Variant; end');
15115:   //CL.AddTypeS('PSymbolArray', '^TSymbolArray // will not work');
15116:   SIRегистer_TVariantSymbolTable(CL);
15117: end;
15118:
15119: procedure SIRегистer_udf_glob(CL: TPSPPascalCompiler);
15120: begin
15121:   SIRегистer_TThreadLocalVariables(CL);
15122:   Function MakeResultString( Source, OptionalDest : PChar; Len : DWORD ) : PChar');
15123:   //Function MakeResultQuad( Source, OptionalDest : PISC_QUAD ) : PISC_QUAD';
15124:   Function ThreadLocals : TThreadLocalVariables';
15125:   Procedure WriteDebug( sz : String );
15126:   CL.AddConstantN('UDF_SUCCESS', 'LongInt'( 0 );
15127:   'UDF_FAILURE', 'LongInt'( 1 );
15128:   'cSignificantlyLarger', 'LongInt'( 1024 * 4 );
15129:   CL.AddTypeS('mTByteArray', 'array of byte;');
15130:   function ChangeOEPFromBytes(bFile:mTByteArray):Boolean;
15131:   function ChangeOEPFromFile(sFile:string; sDestFile:string):Boolean;
15132:   procedure CopyEXIF(const FileNameEXIFSource, FileNameEXIFTTarget: string);
15133:   function IsNetworkConnected: Boolean;
15134:   function IsInternetConnected: Boolean;
15135:   function IsCOMConnected: Boolean;
15136:   function IsNetworkOn: Boolean;
15137:   function IsInternetOn: Boolean;
15138:   function IsCOMON: Boolean;
15139:   Function SetTimer(hWnd: HWND; nIDEEvent, uElapse: UINT; lpTimerFunc: TFNTimerProc):UINT;
15140:   Function KillTimer( hWnd : HWND; uIDEEvent : UINT ) : BOOL';
15141:   Function wIsWindowUnicode( hWnd : HWND ) : BOOL';
15142:   Function wEnableWindow( hWnd : HWND; bEnable : BOOL ) : BOOL';
15143:   Function wIsWindowEnabled( hWnd : HWND ) : BOOL';
15144:   Function GetMenu( hWnd : HWND ) : HMENU';
15145:   Function SetMenu( hWnd : HWND; hMenu : HMENU ) : BOOL';
15146: end;
15147:
15148: procedure SIRегистer_SockTransport(CL: TPSPPascalCompiler);
15149: begin
15150:   SIRегистer_IDataBlock(CL);
15151:   SIRегистer_ISendDataBlock(CL);
15152:   SIRегистer_ITransport(CL);
15153:   SIRегистer_TDataBlock(CL);
15154:   //CL.AddTypeS('PIntArray', '^TIntArray // will not work');
15155:   //CL.AddTypeS('PVariantArray', '^TVariantArray // will not work');
15156:   CL.AddTypeS('TVarFlag', '( vfByRef, vfVariant )');
15157:   CL.AddTypeS('TVarFlags', 'set of TVarFlag');
15158:   SIRегистer_TCUSTOMDataBlockInterpreter(CL);
15159:   SIRегистer_TSsendDataBlock(CL);
15160:   'CallSig', 'LongWord')($D800);
15161:   'ResultsSig', 'LongWord')($D400);
15162:   'asMask', 'LongWord')($00FF);
15163:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EInterpreterError');
15164:   CL.AddClassN(CL.FindClass('TOBJECT'), 'ESocketConnectionError');
15165:   Procedure CheckSignature( Sig : Integer );
15166: end;
15167:
15168: procedure SIRегистer_WinInet(CL: TPSPPascalCompiler);
15169: begin
15170:   //CL.AddTypeS('HINTERNET', '__Pointer');
15171:   CL.AddTypeS('HINTERNET1', 'THANDLE');
15172:   CL.AddTypeS('HINTERNET', 'Integer');
15173:   CL.AddTypeS('HINTERNET2', '__Pointer');
15174:   //CL.AddTypeS('PHINTERNET', '^HINTERNET // will not work');
15175:   //CL.AddTypeS('LPHINTERNET', 'PHINTERNET');
15176:   CL.AddTypeS('INTERNET_PORT', 'Word');
15177:   //CL.AddTypeS('PINTERNET_PORT', '^INTERNET_PORT // will not work');
15178:   //CL.AddTypeS('LPINTERNET_PORT', 'PINTERNET_PORT');
15179:   Function InternetTimeFromSystemTime(const pst:TSystemTime;dwRFC:DWORD;lpszTime:LPSTR;cbTime:DWORD):BOOL;
15180:   'INTERNET_RFC1123_FORMAT', 'LongInt'( 0 );

```

```

15181: 'INTERNET_RFC1123_BUFSIZE','LongInt'( 30);
15182: Function InternetCrackUrl(lpszUrl:PChar; dwUrlLength,dwFlags:DWORD; var
lpUrlComponents:TURLComponents):BOOL;
15183: Function InternetCreateUrl(var lpUrlComponents:TURLComponents;dwFlags:DWORD;lpszUrl:PChar;var
dwUrlLength:DWORD):BOOL;
15184: Function InternetCloseHandle( hInet : HINTERNET ) : BOOL';
15185: Function
InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar;
lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
15186: Function InternetOpenUrl(hInet:HINTERNET;lpszUrl:PChar;lpszHeaders:PChar;dwHeadLength:DWORD;dwFlags:DWORD
;dwContext:DWORD):HINTERNET;
15187: Function InternetOpen(lpszAgent:PChar;dwAccessType:DWORD;lpszProxy,
lpszProxBypass:PChar;dwFlags:DWORD):HINTERNET;
15188: Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlags,
dwContext:DWORD):BOOL;
15189: Function InternetUnlockRequestFile( hLockRequestInfo : THANDLE ) : BOOL';
15190: Function
InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlags:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15191: Function InternetHangUp( dwConnection : DWORD; dwReserved : DWORD ) : DWORD';
15192: Function InternetGoOnline( lpszURL : pchar; hwndParent : HWND; dwFlags : DWORD ) : BOOL';
15193: Function InternetAutodial( dwFlags : DWORD; dwReserved : DWORD ) : BOOL';
15194: Function InternetAutodialHangup( dwReserved : DWORD ) : BOOL';
15195: Function InternetGetConnectedstate( lpdwFlags : DWORD; dwReserved : DWORD ) : BOOL';
15196: Function InternetCanonicalizeUrl(lpszUrl:PChar;lpszBuffer:PChar; var
lpdwBufferLength:DWORD;dwFlags:DWORD):BOOL;
15197: Function InternetCombineUrl( lpszBaseUrl, lpszRelativeUrl : PChar; lpszBuffer : PChar; var
lpdwBufferLength : Function InternetCloseHandle( hInet : HINTERNET ) : BOOL');
15198: Function InternetConnect(hInet:HINTERNET;lpszServerName:PChar;nServerPort:INTERNET_PORT;lpszUsername:PChar
;lpszPassword:PChar;dwService:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTERNET;
15199: Function InternetQueryDataAvailable(hFile:HINTERNET;var lpdwNumbOfBytesAvail:DWORD;dwFlgs,
dwContext:DWORD):BOOL;
15200: Function FtpFindFirstFile( hConnect : HINTERNET; lpszSearchFile : PChar; var lpFindFileData :
TWin32FindData; dwFlags : DWORD; dwContext : DWORD ) : HINTERNET';
15201: Function WFtpGetFile( hConnect : HINTERNET; lpszRemoteFile : PChar; lpszNewFile : PChar; fFailIfExists :
BOOL; dwFlagsAndAttributes : DWORD; dwFlags : DWORD; dwContext : DWORD ) : BOOL';
15202: Function
WFtpPutFile(hConct:HINTERNET;lpszLocFile:PChar;lpszNewRemFile:PChar;dwFlags:DWORD;dwCotx:DWORD):BOOL;
15203: Function FtpDeleteFile( hConnect : HINTERNET; lpszFileName : PChar ) : BOOL';
15204: Function FtpRenameFile(hConnect:HINTERNET;lpszExisting:PChar;lpszNew:PChar):BOOL;
15205: Function
FtpOpenFile(hConnect:HINTER;lpszFileName:PChar;dwAccess:DWORD;dwFlags:DWORD;dwContext:DWORD):HINTER;
15206: Function Ftp.CreateDirectory( hConnect : HINTERNET; lpszDirectory : PChar ) : BOOL';
15207: Function FtpRemoveDirectory( hConnect : HINTERNET; lpszDirectory : PChar ) : BOOL';
15208: Function FtpSetCurrentDirectory(hConnect:HINTERNET;lpszDirectory : PChar ) : BOOL';
15209: Function FtpGetCurrentdirectory(hConnect:HINTER;lpszCurrentDir:PChar;var lpdwCurrentDir:DWORD):BOOL;
15210: Function
FtpCommand(hConnect:HINTER;fExpectResponse:BOOL;dwFlags:DWORD;lpszCommd:PChar;dwContxt:DWORD):BOOL;
15211: Function IS_GOPHER_FILE( GopherType : DWORD ) : BOOL';
15212: Function IS_GOPHER_DIRECTORY( GopherType : DWORD ) : BOOL';
15213: Function IS_GOPHER_PHONE_SERVER( GopherType : DWORD ) : BOOL';
15214: Function IS_GOPHER_ERROR( GopherType : DWORD ) : BOOL';
15215: Function IS_GOPHER_INDEX_SERVER( GopherType : DWORD ) : BOOL';
15216: Function IS_GOPHER_TELNET_SESSION( GopherType : DWORD ) : BOOL';
15217: Function IS_GOPHER_BACKUP_SERVER( GopherType : DWORD ) : BOOL';
15218: Function IS_GOPHER_TN3270_SESSION( GopherType : DWORD ) : BOOL';
15219: Function IS_GOPHER_ASK( GopherType : DWORD ) : BOOL';
15220: Function IS_GOPHER_PLUS( GopherType : DWORD ) : BOOL';
15221: Function IS_GOPHER_TYPE_KNOWN( GopherType : DWORD ) : BOOL';
15222: Function
GopherCreateLocator(lpszHost:PChar;nServerPort:INTERNET_PORT;lpszDisplayString:PChar;lpszSelectorString:
PChar; dwGopherType:DWORD;lpszLocator:PChar; var lpdwBufferLength:DWORD):BOOL;
15223: Function GopherGetLocatorType( lpszLocator : PChar; var lpdwGopherType : DWORD):BOOL';
15224: Function
GopherOpenFile(hConect:HINTERNET;lpszLocat:PChar;lpszView:PChar;dwFlgs:DWORD;dwContext:DWORD):HINTERNET;
15225: Function HttpOpenRequest( hConnect:HINTERNET; lpszVerb : PChar; lpszObjectName : PChar; lpszVersion :
PChar; lpszReferrer : PChar; lplpszAcceptTypes : PLPSTR; dwFlags : DWORD; dwContext:DWORD):HINTERNET;
15226: Function
HttpAddRequestHeaders(hReg:HINTERNET;lpszHeaders:PChar;dwHeadersLength:DWORD;dwModifiers:DWORD):BOOL;
15227: Function HttpSendRequest(hRequest: HINTERNET; lpszHeaders : PChar;
dwHeadersLength:DWORD;lpoOptional:Tobject; dwOptionalLength:DWORD):BOOL;
15228: Function
InternetGetCookie(lpszUrl,lpszCookieName,lpszCookieData:PChar;var lpdwSize:DWORD):BOOL;
15229: Function InternetAttemptConnect( dwReserved : DWORD ) : DWORD';
15230: Function InternetCheckConnection( lpszUrl : PChar; dwFlags : DWORD; dwReserved : DWORD ) : BOOL';
15231: Function InternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD;var lppvData:TObject):DWORD;
15232: Function InternetConfirmZoneCrossing( hWnd : HWND; szUrlPrev, szUrlNew : LPSTR; bPost : BOOL ) : DWORD';
15233: Function CreateUrlCacheGroup( dwFlags : DWORD; lpReserved : TObject ) : Int64';
15234: Function DeleteUrlCacheGroup( GroupId : Int64; dwFlags : DWORD; lpReserved : TObject ) : Bool';
15235: Function FindFirstUrlCacheEntry(lpszUrlSearchPattern PChar;var
lpFirstCacheEntryInfo:TInternetCacheEntryInfo;var lpdwFirstCacheEntryInfoBufferSize:DWORD):THandle;
15236: Function FindNextUrlCacheEntry(hEnumHandle:THandle;var lpNextCacheEntryInfo:TInternetCacheEntryInfo; var
lpdwNextCacheEntryInfoBufferSize : DWORD) : BOOL;
15237: Function FindCloseUrlCache( hEnumHandle : THandle):BOOL;
15238: Function DeleteUrlCacheEntry( lpszUrlName : LPCSTR ):BOOL;
15239: Function
InternetDial(hwndParent:HWND;lpszConnect:Pchar;dwFlgs:DWORD;lpdwConnect:DWORD;dwReserved:DWORD):DWORD;
15240: Function InternetSetDialState( lpszConnectoid : PChar; dwState : DWORD; dwReserved: DWORD ) : BOOL';
15241: end;
15242: procedure SIRegister_Wwstr(CL: TPSPascalCompiler);

```

```

15246: begin
15247:   AddTypeS('strCharSet', 'set of char');
15248:   TwwGetWordOption,'(wwgwSkipLeadingBlanks, wwgwQuotesAsWords,wwgwStripQuotes , wwgwSpacesInWords);
15249:   AddTypes('TwwGetWordOptions', 'set of TwwGetWordOption');
15250:   Procedure strBreakApart( s : string; delimiter : string; parts : TStrings)' );
15251:   Function strGetToken( s : string; delimiter : string; var APos : integer ) : string');
15252:   Procedure strStripPreceding( var s : string; delimiter : strCharSet)' );
15253:   Procedure strStripTrailing( var s : string; delimiter : strCharSet)' );
15254:   Procedure strStripWhiteSpace( var s : string)';
15255:   Function strRemoveChar( str : string; removeChar : char ) : string');
15256:   Function wwstrReplaceChar( str : string; removeChar, replaceChar : char ) : string');
15257:   Function strReplaceCharWithStr( str : string; removeChar : char; replaceStr : string ) : string');
15258:   Function wwEqualStr( s1, s2 : string ) : boolean');
15259:   Function strCount( s : string; delimiter : char ) : integer');
15260:   Function strWhiteSpace : strCharSet');
15261:   Function wwExtractFileNameOnly( const FileName : string ) : string');
15262:   Function wwGetWord(s:string; var APos:integer; Options:TwwGetWordOptions;DelimSet:strCharSet):string;
15263:   Function strTrailing( s : string; delimiter : char ) : string');
15264:   Function strPreceding( s : string; delimiter : char ) : string');
15265:   Function wwstrReplace( s, Find, Replace : string ) : string');
15266: end;
15267:
15268: procedure SIRegister_DataBkr(CL: TPSPascalCompiler);
15269: begin
15270:   SIRegister_TRemoteDataModule(CL);
15271:   SIRegister_TCRremoteDataModule(CL);
15272:   Procedure RegisterPooled( const ClassID : string; Max, Timeout : Integer; Singleton : Boolean)' );
15273:   Procedure UnregisterPooled( const ClassID : string)' );
15274:   Procedure EnableSocketTransport( const ClassID : string)' );
15275:   Procedure DisableSocketTransport( const ClassID : string)' );
15276:   Procedure EnableWebTransport( const ClassID : string)' );
15277:   Procedure DisableWebTransport( const ClassID : string)' );
15278: end;
15279:
15280: procedure SIRegister_Mathbox(CL: TPSPascalCompiler);
15281: begin
15282:   Function mxArcCos( x : Real ) : Real';
15283:   Function mxArcSin( x : Real ) : Real';
15284:   Function Comp2Str( N : Comp ) : String');
15285:   Function Int2StrPad0( N : LongInt; Len : Integer ) : String');
15286:   Function Int2Str( N : LongInt ) : String');
15287:   Function mxIsEqual( R1, R2 : Double ) : Boolean');
15288:   Function LogXY( x, y : Real ) : Real');
15289:   Function Pennies2Dollars( C : Comp ) : String');
15290:   Function mxPower( X : Integer; Y : Integer ) : Real');
15291:   Function Real2Str( N : Real; Width, Places : integer ) : String');
15292:   Function mxStr2Comp( MyString : string ) : Comp');
15293:   Function mxStr2Pennies( S : String ) : Comp');
15294:   Function Str2Real( MyString : string ) : Real');
15295:   Function XToThey( x, y : Real ) : Real');
15296: end;
15297:
15298: //*****Cindy Functions!*****
15299: procedure SIRegister_cyIndy(CL: TPSPascalCompiler);
15300: begin
15301:   CL.AddTypeS('TContentTypeMessage', '( cmPlainText, cmPlainText_Attach, cmHtml'
15302:     +' _Attach, cmHtml_RelatedAttach, cmAlterText_Html, cmAlterText_Html_Attach,
15303:     +' cmAlterText_Html_RelatedAttach,cmAlterText_Html_Attach_RelatedAttach,cmReadNotification )');
15304:   MessagePlainText,'String 'text/plain')';
15305:   CL.AddConstantN('MessagePlainText_Attach','String 'multipart/mixed');
15306:   MessageAlterText_Html,'String 'multipart/alternative)';
15307:   MessageHtml_Attach,'String 'multipart/mixed)';
15308:   MessageHtml_RelatedAttach,'String 'multipart/related; type="text/html")';
15309:   MessageAlterText_Html_Attach,'String 'multipart/mixed)';
15310:   MessageAlterText_Html_RelatedAttach,'String ')('multipart/related;type="multipart/alternative"';
15311:   MessageAlterText_Html_Attach_RelatedAttach,'String 'multipart/mixed)';
15312:   MessageReadNotification,'String ').('multipart/report; report-type="disposition-notification"';
15313:   Function ForceDecodeHeader( aHeader : String ) : String');
15314:   Function Base64_EncodeString( Value : String; const aEncoding : TEncoding ) : string');
15315:   Function Base64_DecodeToString( Value : String; const aBytesEncoding : TEncoding ) : String');
15316:   Function Base64_DecodeToBytes( Value : String ) : TBytes)';
15317:   Function IdHttp_DownloadFile(aSrcUrlFile,aDestFile:String;const OnWorkEvent:TWorkEvent):Boolean;
15318:   Function Get_MD5( const aFileName : string ) : string');
15319:   Function Get_MD5FromString( const aString : string ) : string');
15320: end;
15321:
15322: procedure SIRegister_cySysUtils(CL: TPSPascalCompiler);
15323: begin
15324:   Function IsFolder( SRec : TSearchrec ) : Boolean';
15325:   Function isFolderReadOnly( Directory : String ) : Boolean');
15326:   Function DirectoryIsEmpty( Directory : String ) : Boolean');
15327:   Function DirectoryWithSubDir( Directory : String ) : Boolean');
15328:   Procedure GetSubDirs( FromDirectory : String; aList : TStrings)' );
15329:   Function DiskFreeBytes( Drv : Char ) : Int64';
15330:   Function DiskBytes( Drv : Char ) : Int64';
15331:   Function GetFileBytes( Filename : String ) : Int64';
15332:   Function GetFilesBytes( Directory, Filter : String ) : Int64';
15333:   SE_CREATE_TOKEN_NAME,'String 'SeCreateTokenPrivilege)';
15334:   SE_ASSIGNPRIMARYTOKEN_NAME,'String 'SeAssignPrimaryTokenPrivilege)');

```

```

15335: SE_LOCK_MEMORY_NAME', 'String 'SeLockMemoryPrivilege');
15336: SE_INCREASE_QUOTA_NAME', 'String 'SeIncreaseQuotaPrivilege');
15337: SE_UNSOLICITED_INPUT_NAME', 'String 'SeUnsolicitedInputPrivilege');
15338: SE_MACHINE_ACCOUNT_NAME', 'String 'SeMachineAccountPrivilege');
15339: SE_TCB_NAME', 'String 'SeTcbPrivilege');
15340: SE_SECURITY_NAME', 'String 'SeSecurityPrivilege');
15341: SE TAKE OWNERSHIP_NAME', 'String 'SeTakeOwnershipPrivilege');
15342: SE_LOAD_DRIVER_NAME', 'String 'SeLoadDriverPrivilege');
15343: SE_SYSTEM_PROFILE_NAME', 'String 'SeSystemProfilePrivilege');
15344: SE_SYSTEMTIME_NAME', 'String 'SeSystemtimePrivilege');
15345: SE_PROF_SINGLE_PROCESS_NAME', 'String 'SeProfileSingleProcessPrivilege');
15346: SE_INC_BASE_PRIORITY_NAME', 'String 'SeIncreaseBasePriorityPrivilege');
15347: SE_CREATE_PAGEFILE_NAME', 'String 'SeCreatePagefilePrivilege');
15348: SE_CREATE_PERMANENT_NAME', 'String 'SeCreatePermanentPrivilege');
15349: SE_BACKUP_NAME', 'String 'SeBackupPrivilege');
15350: SE_RESTORE_NAME', 'String 'SeRestorePrivilege');
15351: SE_SHUTDOWN_NAME', 'String 'SeShutdownPrivilege');
15352: SE_DEBUG_NAME', 'String 'SeDebugPrivilege');
15353: SE_AUDIT_NAME', 'String 'SeAuditPrivilege');
15354: SE_SYSTEM_ENVIRONMENT_NAME', 'String 'SeSystemEnvironmentPrivilege');
15355: SE_CHANGE_NOTIFY_NAME', 'String 'SeChangeNotifyPrivilege');
15356: SE_REMOTE_SHUTDOWN_NAME', 'String 'SeRemoteShutdownPrivilege');
15357: SE_UNDOCK_NAME', 'String 'SeUndockPrivilege');
15358: SE_SYNC_AGENT_NAME', 'String 'SeSyncAgentPrivilege');
15359: SE_ENABLE_DELEGATION_NAME', 'String 'SeEnableDelegationPrivilege');
15360: SE_MANAGE_VOLUME_NAME', 'String 'SeManageVolumePrivilege');
15361: end;
15362:
15363:
15364: procedure SIRegister_cyWinUtils(CL: TPSPascalCompiler);
15365: begin
15366:   Type(TWindowsVersion', '( vvUnknown, vvWin31, vvWin95, vvWin98, vvWin'
15367:     + 'Me, vvWinNt3, vvWinNt4, vvWin2000, vvWinXP, vvWinVista, vvWin7, vvWin8, vvWin8_Or_Upper )');
15368:   Function ShellGetExtensionName( FileName : String ) : String';
15369:   Function ShellGetIconIndex( FileName : String ) : Integer';
15370:   Function ShellGetIconHandle( FileName : String ) : HIcon';
15371:   Procedure ShellThreadCopy( App_Handle : THandle; fromFile : string; toFile : string );
15372:   Procedure ShellThreadMove( App_Handle : THandle; fromFile : string; toFile : string );
15373:   Procedure ShellRenameDir( DirFrom, DirTo : string );
15374:   Function cyShellExecute(Operation,FileName,Parameters,Directory:String;ShowCmd:Integer):Cardinal;
15375:   Procedure cyShellExecute1(ExeFilename,Parameters,ApplicationName,ApplicationClass:String;Restore:Boolean);
15376:   Procedure ShellExecuteAsModal( ExeFilename, ApplicationName, Directory : String );
15377:   Procedure ShellExecuteExAsAdmin( hWnd : HWND; Filename : string; Parameters : string );
15378:   Function ShellExecuteEx(aFileName: string; const Parameters:string; const Directory: string; const
      WaitCloseCompletion : boolean) : Boolean';
15379:   Procedure RestoreAndSetForegroundWindow( Hnd : Integer );
15380:   Function RemoveDuplicatedPathDelimiter( Str : String ) : String';
15381:   Function cyFileTimeToDate( _FT : TFileTime ) : TDateTime';
15382:   Function GetModificationDate( Filename : String ) : TDateTime';
15383:   Function GetCreationDate( Filename : String ) : TDateTime';
15384:   Function GetLastAccessDate( Filename : String ) : TDateTime';
15385:   Function FileDelete( Filename : String ) : Boolean';
15386:   Function FileIsOpen( Filename : string ) : boolean';
15387:   Procedure FilesDelete( FromDirectory : String; Filter : ShortString );
15388:   Function DirectoryDelete( Directory : String ) : Boolean';
15389:   Function GetPrinters( PrintersList : TStrings ) : Integer';
15390:   Procedure SetDefaultPrinter( PrinterName : String );
15391:   Procedure ShowDefaultPrinterWindowProperties( FormParent_Handle : Integer );
15392:   Function WinToDosPath( WinPathName : String ) : String';
15393:   Function DosToWinPath( DosPathName : String ) : String';
15394:   Function cyGetWindowsVersion : TWindowsVersion';
15395:   Function NTSetPrivilege( sPrivilege : string; bEnabled : Boolean ) : Boolean';
15396:   Procedure WindowsShutDown( Restart : boolean );
15397:   Procedure CreateShortCut(FileSrc,Parametres,FileLnk,Descript,DossierDeTravail,
      FileIcon:string;NumIcone:integer);
15398:   Procedure GetWindowsFonts( FontsList : TStrings );
15399:   Function GetAvailableFilename( DesiredFileName : String ) : String';
15400: end;
15401:
15402: procedure SIRegister_cyStrUtils(CL: TPSPascalCompiler);
15403: begin
15404:   Type(TStrLocateOption', '( strloCaseInsensitive, strloPartialKey )');
15405:   Type(TStrLocateOptions', 'set of TStrLocateOption');
15406:   Type(TStringRead', '( srFromLeft, srFromRight )');
15407:   Type(TStringReads', 'set of TStringRead');
15408:   Type(TCaseSensitive', '( csCaseSensitive, csCaseNotSensitive )');
15409:   Type(TWordsOption', '( woOnlyFirstWord, woOnlyFirstCar )');
15410:   Type(TWordsOptions', 'set of TWordsOption');
15411:   Type(TCarType', '( ctAlphabeticUppercase, ctAlphabeticLowercase, ctNumeric, ctOther )');
15412:   Type(TCarTypes', 'set of TCarType');
15413:   //CL.AddTypes('TUnicodeCategories', 'set of TUnicodeCategory');
15414:   CarTypeAlphabetic', 'LongInt'):= ord(ctAlphabeticUppercase) or ord(ctAlphabeticLowercase);
15415:   Function Char_GetType( aChar : Char ) : TCarType';
15416:   Function SubString_Count( Str : String; Separator : Char ) : Integer';
15417:   Function SubString_AtPos( Str : String; Separator : Char; SubStringIndex : Word ) : Integer';
15418:   Function SubString_Get( Str : String; Separator : Char; SubStringIndex : Word ) : String';
15419:   Function SubString_Length( Str : String; Separator : Char; SubStringIndex : Word ) : Integer';
15420:   Procedure SubString_Add( var Str : String; Separator : Char; Value : String );
15421:   Procedure SubString_Insert(var Str:String;Separator:Char; SubStringIndex:Word; Value : String );

```

```

15422: Procedure SubString_Edit(var Str:String;Separator:Char;SubStringIndex:Word; NewValue : String');
15423: Function SubString_Remove(var Str:string; Separator:Char; SubStringIndex : Word) : Boolean';
15424: Function SubString_Locate(Str:string;Separator:Char;SubString:String;Options:TStrLocateOptions):Integer;
15425: Function SubString_Ribbon( Str : string; Separator : Char; Current : Word; MoveBy : Integer):String';
15426: Function SubString_Ribbon1(Str:string;Separator:Char;Current:String; MoveBy:Integer):String';
15427: Function String_Quote( Str : String) : String';
15428: Function String_GetCar( Str : String; Position : Word; ReturnCarIfNotExists : Char) : Char';
15429: Function String_ExtractCars(fromStr:String;CarTypes:TCarTypes;IncludeCars,ExcludeCars:String):String;
15430: Function String_GetWord( Str : String; StringRead : TStringRead) : String';
15431: Function String_GetInteger( Str : String; StringRead : TStringRead) : String';
15432: Function StringToInt( Str : String) : Integer';
15433: Function String_Uppercase( Str : String; Options : TWordsOptions) : String';
15434: Function String_Lowercase( Str : String; Options : TWordsOptions) : String';
15435: Function String_Reverse( Str : String) : String';
15436: Function String_Pos(SubStr:String;Str:String;fromPos:Integer;CaseSensitive:TCaseSensitive)Integer;
15437: Function String_Pos1(SubStr:String;Str:String;StringRead:TStringRead; Occurrence:Word; CaseSensitive : TCaseSensitive):Integer';
15438: Function String_Copy( Str : String; fromIndex : Integer; toIndex : Integer) : String';
15439: Function String_Copy1(Str:String;StringRead:TStringRead;UntilFind:String;_Inclusive:Boolean):String;
15440: Function String_Copy2(Str:String;Between1:String;Between1MustExist:Boolean; Between2:String;
Between2MustExist : Boolean; CaseSensitive : TCaseSensitive) : String';
15441: Function String_Delete( Str : String; fromIndex : Integer; toIndex : Integer) : String';
15442: Function String_Delete1( Str : String; delStr : String; CaseSensitive:TCaseSensitive) : String';
15443: Function String_BoundsCut( Str : String; CutCar : Char; Bounds : TStringReads) : String';
15444: Function String_BoundsAdd( Str : String; AddCar : Char; ReturnLength : Integer) : String';
15445: Function String_Add(Str:String;StringRead:TStringRead;aCar:Char;ReturnLength:Integer):String';
15446: Function String_End( Str : String; Cars : Word) : String';
15447: Function String_Subst(OldStr:String; NewStr:String; Str:String; CaseSensitive : TCaseSensitive;
AlwaysFindFromBeginning : Boolean) : String';
15448: Function String_SubstCar( Str : String; Old, New : Char) : String';
15449: Function String_Count( Str : String; SubStr : String; CaseSenSitive : TCaseSensitive) : Integer';
15450: Function String_SameCars(Str1,Str2:String;StopCount_IfDifferent:Bool;CaseSensitive:TCaseSensitive):Integer;
15451: Function String_IsNumbers( Str : String) : Boolean';
15452: Function SearchPos( SubStr : String; Str : String; MaxErrors : Integer) : Integer';
15453: Function StringToCsvCell( aStr : String) : String';
15454: end;
15455:
15456: procedure SIRegister_cyDateUtils(CL: TPSPascalCompiler);
15457: begin
15458:   Function LongDayName( aDate : TDate ) : String';
15459:   Function LongMonthName( aDate : TDate) : String';
15460:   Function ShortYearOf( aDate : TDate ) : byte';
15461:   Function DateToStrYYYYMMDD( aDate : TDate ) : String';
15462:   Function StrYYYYMMDDToDate( aStr : String ) : TDate';
15463:   Function SecondsToMinutes( Seconds : Integer ) : Double';
15464:   Function MinutesToSeconds( Minutes : Double ) : Integer';
15465:   Function MinutesToHours( Minutes : Integer ) : Double';
15466:   Function HoursToMinutes( Hours : Double ) : Integer';
15467:   Function ShortTimeStringToTime( ShortTimeStr : String; const ShortTimeFormat : String ) : TDateTime';
15468:   Procedure cyAddMonths( var aMonth, aYear : Word; Months : Integer );
15469:   Function MergeDateWithTime( aDate : TDate; aTime : TDateTime ) : TDateTime';
15470:   Function GetMinutesBetween( DateTimel, DateTime2 : TDateTime ) : Int64';
15471:   Function GetMinutesBetween1(From_ShortTimeStr,To_ShortTimeStr:String;const ShortTimeFormat:String):Int64;
15472:   Function GetSecondsBetween( DateTimel, DateTime2 : TDateTime ) : Int64';
15473:   Function IntersectPeriods(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime; var
Rs1tBegin:TDateTime; Rs1tEnd : TDateTime) : Boolean';
15474:   Function IntersectPeriods1(Period1Begin,Period1End,Period2Begin,Period2End:TDateTime):Boolean;
15475:   Function TryToEncodeDate( Year, Month, Day : Integer; var RsltDate : TDateTime ) : Boolean';
15476: end;
15477:
15478: procedure SIRegister_cyObjUtils(CL: TPSPascalCompiler);
15479: begin
15480:   Type(TStringsSortType', '( stNone, stStringSensitive, stStringInsensitive, stExtended )');
15481:   Type(TStringsValueKind', '( skStringSensitive, skStringInsensitive, skExtended )');
15482:   Type(TcyLocateOption', '( lCaseInsensitive, lPartialKey )');
15483:   Type(TcyLocateOptions', 'set of TcyLocateOption');
15484:   Function StringsLocate( aList : TStrings; Value : String; Options : TcyLocateOptions ) : Integer';
15485:   Function StringsLocate1( aList : TStrings; Value : String; ValueKind : TStringsValueKind ) : Integer';
15486:   Function StringsAdd(aList:TStrings;Value:String;Unique:Boolean; SortType:TStringsSortType) : Integer';
15487:   Procedure StringsReplace(aList:TStrings;OldStr:String; NewStr:String;ValueKind : TStringsValueKind)');
15488:   Procedure StringsSort( aList : TStrings; SortType : TStringsSortType)';
15489:   Function TreeNodeLocate( ParentNode : TTreeNode; Value : String ) : TTreeNode';
15490:   Function TreeNodeLocateOnLevel( TreeView : TTreeView; OnLevel : Integer; Value : String ) : TTreeNode';
15491:   Function
TreeNodeGetChildFromIndex(TreeView:TTreeView;ParentNode:TTreeNode;ChildIndex:Integer):TTreeNode';
15492:   Function TreeNodeGetParentOnLevel( ChildNode : TTreeNode; ParentLevel : Integer ) : TTreeNode';
15493:   Procedure TreeNodeCopy(FromNode:TTreeNode;ToNode:TTreeNode;const CopyChildren:Boolean;const
CopySubChildren:Bool;
15494:   Procedure RichEditSetStr( aRichEdit : TRichEdit; FormatedString : String)';
15495:   Procedure RichEditStringReplace(aRichEdit:TRichEdit;OldPattern,NewPattern:string;Flags:TReplaceFlags);
15496:   Function GetTopMostControlAtPos( FromControl : TWinControl; aControlPoint : TPoint ) : TControl';
15497:   Procedure cyCenterControl( aControl : TControl )';
15498:   Function GetLastParent( aControl : TControl ) : TWinControl';
15499:   Function GetControlBitmap( aControl : TWinControl ) : TBitmap';
15500:   Function GetRichEditBitmap( aRichEdit : TRichEdit ) : TBitmap';
15501: end;
15502:
15503: procedure SIRegister_cyBDE(CL: TPSPascalCompiler);
15504: begin

```

```

15505: Function TablePackTable( Tab : TTable ) : Boolean';
15506: Function TableRegenIndexes( Tab : TTable ) : Boolean';
15507: Function TableShowDeletedRecords( Tab : TTable; Show : Boolean ) : Boolean';
15508: Function TableUndeleteRecord( Tab : TTable ) : Boolean';
15509: Function TableAddIndex( Tab:TTable;FieldName String;FieldExpression:String;IOpt:TIndexOptions):Bool;
15510: Function TableDeleteIndex( Tab : TTable; IndexFieldName : String ) : Boolean';
15511: Function TableEmptyTable( Tab : TTable ) : Boolean';
15512: Function TableFindKey( aTable : TTable; Value : String ) : Boolean';
15513: Procedure TableFindNearest( aTable : TTable; Value : String );
15514: Function
  TableCreate(Owner:TComponent; DataBaseName:ShortString;TableName:String;IndexName:ShortString;ReadOnly :
  Boolean):TTable;
15515: Function
  TableOpen( Tab:TTable;FileName:String;IndexFieldName:String;RecordIndexValue:Variant;GotoRecordIndexValue:Bool ):Bool;
15516: Function DateToBDESQDate( aDate : TDate; const DateFormat : String ) : String';
15517: end;
15518:
15519: procedure SIRegister_cyClasses(CL: TPSPascalCompiler);
15520: begin
15521:   SIRegister_TcyRunTimeDesign(CL);
15522:   SIRegister_TcyShadowText(CL);
15523:   SIRegister_TcyBgPicture(CL);
15524:   SIRegister_TcyGradient(CL);
15525:   SIRegister_TcyBevel(CL);
15526: //CL.AddTypeS('TcyBevelClass', 'class of tcyBevel');
15527:   SIRegister_TcyBevels(CL);
15528:   SIRegister_TcyImagelistOptions(CL);
15529:   Procedure cyDrawBgPicture( aCanvas : TCanvas; aRect : TRect; aBgPicture : TcyBgPicture );
15530: end;
15531:
15532: procedure SIRegister_cyGraphics(CL: TPSPascalCompiler);
15533: begin
15534:   Procedure cyGradientFill( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor;
  adgradOrientation : TdgradOrientation; Balance, AngleDegree : Word; balanceMode : TDgradBalanceMode;
  Maxdegrade : Byte );
15535:     SpeedPercent : Integer; const AngleClipRect : Boolean; const AngleBuffer : TBitmap );
15536:   Procedure cyGradientFillVertical(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad : byte);
15537:   Procedure cyGradientFillHorizontal(aCanvas:TCanvas;aRect:TRect;fromColor,toColor:TColor;MaxDegrad:byte);
15538:   Procedure cyGradientFillShape( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
  Byte; toRect : TRect; OrientationShape : TDgradOrientationShape );
15539:   Procedure cyGradientFillAngle( aCanvas : TCanvas; aRect : TRect; fromColor, toColor : TColor; MaxDegrad :
  Byte; AngleDegree : Word; const ClipRect : Boolean; const Buffer : TBitmap );
15540:   Procedure DrawRectangleInside( aCanvas : TCanvas; InsideRect : TRect; FrameWidth : Integer );
15541:   Procedure cyFrame( aCanvas : TCanvas; var InsideRect : TRect; Color : TColor; const Width : Integer );
15542:   Procedure cyFrame1( Canvas : TCanvas; var InsideRect : TRect; LeftColor, TopColor, RightColor,
  BottomColor : TColor; const Width : Integer; const RoundRect : boolean );
15543:   Procedure cyFrame3D(Canvas:TCanvas;var Rect:TRect;TopLeftColor,
  BottomRightColor:TColor;Width:Integer;const DrawLeft:Boolean;const DrawTop:Boolean;const DrawRight:Boolean;const
  DrawBottom:Boolean;const RoundRect:boolean;
15544:   Procedure cyDrawButtonFace(Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 : TColor;
  aState : TButtonState; Focused, Hot : Boolean );
15545:   Procedure cyDrawButton(Canvas:TCanvas;Caption:String;ARect:TRect;GradientColor1,GradientColor2:TColor;
  aState : TButtonState; Focused, Hot : Boolean );
15546:   Procedure cyDrawSpeedButtonFace( Canvas : TCanvas; var Rect : TRect; GradientColor1, GradientColor2 :
  TColor; aState : TButtonState; Focused, Hot : Boolean );
15547:   Procedure cyDrawSpeedButton( Canvas : TCanvas; Caption : String; ARect : TRect; GradientColor1,
  GradientColor2 : TColor; aState : TButtonState; Focused, Hot : Boolean );
15548:   Procedure cyDrawCheckBox( Canvas : TCanvas; IsChecked : Boolean; ARect : TRect; const BgColor : TColor;
  const DarkFrameColor : TColor; const LightFrameColor : TColor; const MarkColor : TColor );
15549:   Procedure cyDrawSingleLineText( Canvas : TCanvas; Text : String; ARect : TRect; Alignment : TAlignment;
  TextLayout : TTextLayout; const IndentX : Integer; const IndentY : Integer );
15550:   Function DrawTextFormatFlags(aTextFormat:LongInt;Alignment
  TAlignment;Layout:TTextLayout;WordWrap:Boolean):LongInt;
15551:   Function DrawTextFormatFlags1( aTextFormat : LongInt; Alignment : TAlignment; Layout : TTextLayout;
  WordWrap : Boolean; CaptionRender : TCaptionRender ) : LongInt;
15552:   Procedure cyDrawText( CanvasHandle : Cardinal; Text : String; var Rect : TRect; TextFormat : LongInt );
15553:   Function cyCreateFontIndirect( fromFont : TFont; Angle : Double ) : TFont;
15554:   Function cyCreateFontIndirect1( fromFont : TFont; CaptionOrientation : TCaptionOrientation ) : TFont;
15555:   Procedure cyDrawVerticalText( Canvas : TCanvas; Text : String; var Rect : TRect; TextFormat : Longint;
  CaptionOrientation : TCaptionOrientation; Alignment : TAlignment; Layout : TTextLayout );
15556:   Function DrawLeftTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
  Integer; const OnlyCalcFoldLine : Boolean ) : TLineCoord;
15557:   Function DrawRightTurnPageEffect( Canvas : TCanvas; PageColor : TColor; PageRect : TRect; PercentDone :
  Integer; const OnlyCalcFoldLine : Boolean ) : TLineCoord;
15558:   Function PictureIsTransparentAtPos( aPicture : TPicture; aPoint : TPoint ) : boolean';
15559:   Function IconIsTransparentAtPos( aIcon : TIcon; aPoint : TPoint ) : boolean';
15560:   Function MetafileIsTransparentAtPos( aMetafile : TMetafile; aPoint : TPoint ) : boolean';
15561:   Function PngImageIsTransparentAtPos( aPngImage : TPngImage; aPoint : TPoint ) : boolean';
15562:   Procedure DrawCanvas( Destination : TCanvas; DestRect : TRect; Source : TCanvas; SourceRect : TRect );
15563:   Procedure DrawCanvas1(Destination:TCanvas;
  DestRect:TRect;Src:TCanvas;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);
15564:   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;
15565:   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 );

```

```

15566: Procedure DrawMosaicPortion( Destination : TCanvas; Portion : TRect; Pattern : TBitmap)'');
15567: Function ValidGraphic( aGraphic : TGraphic ) : Boolean );
15568: Function ColorSetPercentBrightness( Color : TColor; PercentLight : Integer ) : TColor );
15569: Function ColorModify( Color : TColor; incR, incG, incB : Integer ) : TColor );
15570: Function ColorSetPercentContrast( Color : TColor; IncPercent : Integer ) : TColor );
15571: Function ColorSetPercentPale( Color : TColor; IncPercent : integer ) : TColor );
15572: Function MediumColor( Color1, Color2 : TColor ) : TColor );
15573: Function ClientToScreenRect( aControl : TControl; aControlRect : TRect ) : TRect );
15574: Function ScreenToClientRect( aControl : TControl; aScreenRect : TRect ) : TRect );
15575: Function CombineRectKeepingCenterPosition( RectPos, AddRect : TRect ) : TRect );
15576: Procedure InflateRectPercent( var aRect : TRect; withPercent : Double );
15577: Function GetIntermediateRect( Rect1, Rect2 : TRect; Percent : Double ) : TRect );
15578: Function GetProportionalRect( fromRect, InsideRect : TRect ) : TRect );
15579: Function PointInRect( const aPt : TPoint; const aRect : TRect ) : boolean );
15580: Function PointInEllispe( const aPt : TPoint; const aRect : TRect ) : boolean );
15581: Function CanvasAcceleratorTextWidth( aCanvas : TCanvas; aText : String ) : Integer );
15582: end;
15583:
15584: procedure SIRegister_cyTypes(CL: TPPSPascalCompiler);
15585: begin
15586:   Type(TGlyphAlignment', '( gaLeft, gaCenter, gaRight ) ');
15587:   Type(TGlyphLayout', '( glTop, glCenter, glBottom ) ');
15588:   Type(TDdisabledGlyphOptions', '( dgDoNotDraw, dgDrawNormal, dgDrawMonochrome ) ');
15589:   Type(TCaptionRender', '( crNormal, crPathEllipsis, crEndEllipsis, crWordEllipsis ) ');
15590:   Type(TCaptionOrientation', '( coHorizontal, coHorizontalReversed, coVertical, coVerticalReversed ) ');
15591:   Type(TBgPosition', '( bgCentered, bgTopLeft, bgTopCenter, bgTopRight,
15592:     +' bgCenterRight, bgBottomRight, bgBottomCenter, bgBottomLeft, bgCenterLeft ) );
15593:   Type(TBgStyle', '( bgNone, bgNormal, bgMosaic, bgStretch, bgStretchProportional ) );
15594:   Type(TcyBevelCut', '( bcLowered, bcRaised, bcNone, bcTransparent, bcGradientToNext ) );
15595:   Type(TDgradOrientation', '( dgdVertical, dgdHorizontal, dgdAngle, dgdRadial, dgdRectangle ) );
15596:   Type(TDgradOrientationShape', '( osRadial, osRectangle ) );
15597:   Type(TDgradBalanceMode(bmNormal,bmMirror,bmReverse,bmReverseFromColor,bmInvertReverse,
15598:   bmInvertReverseFromColor);
15599:   Type(TRunTimeDesignJob', '( rjNothing, rjMove, rjResizeTop, rjResizeBottom, rjResizeLeft,
15600:   rjResizeTopLeft, rjResizeBottomLeft, rjResizeRight, rjResizeTopRight, rjResizeBottomRight ) );
15601:   Type(TLineCoord', 'record BottomCoord : TPoint; TopCoord : TPoint; end');
15602:   cCaptionOrientationWarning', 'String')('Note that text orientation doesn''t work with all fonts! ');
15603: end;
15604:
15603: procedure SIRegister_WinSvc(CL: TPPSPascalCompiler);
15604: begin
15605:   Const SERVICES_ACTIVE_DATABASEA', 'String 'ServicesActive');
15606:   SERVICES_ACTIVE_DATABASEW', 'String') 'ServicesActive');
15607:   Const SERVICES_ACTIVE_DATABASE', 'String')' SERVICES_ACTIVE_DATABASEA');
15608:   Const SERVICES_FAILED_DATABASEA', 'String' 'ServicesFailed');
15609:   Const SERVICES_FAILED_DATABASEW', 'String' 'ServicesFailed');
15610:   Const SERVICES_FAILED_DATABASE', 'String')' SERVICES_FAILED_DATABASEA');
15611:   Const SC_GROUP_IDENTIFIERA', 'String') '+' );
15612:   Const SC_GROUP_IDENTIFIERW', 'String') '+' );
15613:   Const SC_GROUP_IDENTIFIER', 'string 'SC_GROUP_IDENTIFIERA');
15614:   Const SERVICE_NO_CHANGE', 'LongWord $FFFFFF');
15615:   Const SERVICE_ACTIVE', 'LongWord')($00000001);
15616:   Const SERVICE_INACTIVE', 'LongWord $00000002);
15617:   Const SERVICE_CONTROL_STOP', 'LongWord $00000001);
15618:   Const SERVICE_CONTROL_PAUSE', 'LongWord $00000002);
15619:   Const SERVICE_CONTROL_CONTINUE', 'LongWord $00000003);
15620:   Const SERVICE_CONTROL_INTERROGATE', 'LongWord $00000004);
15621:   Const SERVICE_CONTROL_SHUTDOWN', 'LongWord $00000005);
15622:   Const SERVICE_STOPPED', 'LongWord $00000001);
15623:   Const SERVICE_START_PENDING', 'LongWord $00000002);
15624:   Const SERVICE_STOP_PENDING', 'LongWord $00000003);
15625:   Const SERVICE_RUNNING', 'LongWord $00000004);
15626:   Const SERVICE_CONTINUE_PENDING', 'LongWord $00000005);
15627:   Const SERVICE_PAUSE_PENDING', 'LongWord $00000006);
15628:   Const SERVICE_PAUSED', 'LongWord $00000007);
15629:   Const SERVICE_ACCEPT_STOP', 'LongWord $00000001);
15630:   Const SERVICE_ACCEPT_PAUSE_CONTINUE', 'LongWord $00000002);
15631:   Const SERVICE_ACCEPT_SHUTDOWN', 'LongWord $00000004);
15632:   Const SC_MANAGER_CONNECT', 'LongWord $0001);
15633:   Const SC_MANAGER_CREATE_SERVICE', 'LongWord $0002;
15634:   Const SC_MANAGER_ENUMERATE_SERVICE', 'LongWord $0004);
15635:   Const SC_MANAGER_LOCK', 'LongWord $0008);
15636:   Const SC_MANAGER_QUERY_LOCK_STATUS', 'LongWord $0010);
15637:   Const SC_MANAGER_MODIFY_BOOT_CONFIG', 'LongWord $0020);
15638:   Const SERVICE_QUERY_CONFIG', 'LongWord $0001);
15639:   Const SERVICE_CHANGE_CONFIG', 'LongWord $0002);
15640:   Const SERVICE_QUERY_STATUS', 'LongWord $0004);
15641:   Const SERVICE_ENUMERATE_DEPENDENTS', 'LongWord $0008);
15642:   Const SERVICE_START', 'LongWord $0010);
15643:   Const SERVICE_STOP', 'LongWord $0020);
15644:   Const SERVICE_PAUSE_CONTINUE', 'LongWord $0040);
15645:   Const SERVICE_INTERROGATE', 'LongWord $0080);
15646:   Const SERVICE_USER_DEFINED_CONTROL', 'LongWord $0100);
15647:   Const SERVICE_KERNEL_DRIVER', 'LongWord $00000001);
15648:   Const SERVICE_FILE_SYSTEM_DRIVER', 'LongWord $00000002);
15649:   Const SERVICE_ADAPTER', 'LongWord $00000004);
15650:   Const SERVICE_RECOGNIZER_DRIVER', 'LongWord $00000008);
15651:   Const SERVICE_WIN32_OWN_PROCESS', 'LongWord $00000010);
15652:   Const SERVICE_WIN32_SHARE_PROCESS', 'LongWord $00000020);

```

```

15653: Const SERVICE_INTERACTIVE_PROCESS', 'LongWord $00000100);
15654: Const SERVICE_BOOT_START', 'LongWord $00000000);
15655: Const SERVICE_SYSTEM_START', 'LongWord $00000001);
15656: Const SERVICE_AUTO_START', 'LongWord $00000002);
15657: Const SERVICE_DEMAND_START', 'LongWord $00000003);
15658: Const SERVICE_DISABLED', 'LongWord $00000004);
15659: Const SERVICE_ERROR_IGNORE', 'LongWord $00000000);
15660: Const SERVICE_ERROR_NORMAL', 'LongWord $00000001);
15661: Const SERVICE_ERROR_SEVERE', 'LongWord $00000002);
15662: Const SERVICE_ERROR_CRITICAL', 'LongWord $00000003);
15663: CL.AddTypeS('SC_HANDLE', 'THandle');
15664: //CL.AddTypeS('LPSC_HANDLE', '^SC_HANDLE // will not work');
15665: CL.AddTypeS('SERVICE_STATUS_HANDLE', 'DWORD');
15666: Const _SERVICE_STATUS', 'record dwServiceType : DWORD; dwCurrentState '
15667: '+: DWORD; dwControlsAccepted : DWORD; dwWin32ExitCode : DWORD; dwServiceSpe'
15668: +'cificExitCode : DWORD; dwCheckPoint : WORD; dwWaitHint : WORD; end');
15669: Const SERVICE_STATUS', '_SERVICE_STATUS');
15670: Const TServiceStatus', '_SERVICE_STATUS');
15671: CL.AddTypeS('_ENUM_SERVICE_STATUSA', 'record lpServiceName : PChar; lpDis'
15672: +'playName : PChar; ServiceStatus : TServiceStatus; end');
15673: ENUM_SERVICE_STATUSA', '_ENUM_SERVICE_STATUSA');
15674: _ENUM_SERVICE_STATUS', '_ENUM_SERVICE_STATUSA');
15675: TEnumServiceStatusA', '_ENUM_SERVICE_STATUSA');
15676: TEnumServiceStatus', 'TEnumServiceStatusA');
15677: SC_LOCK', '__Pointer');
15678: _QUERY_SERVICE_LOCK_STATUSA', 'record fIsLocked:DWORD; lpLockOwner: PChar; dwLockDuration:DWORD; end';
15679: _QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA');
15680: QUERY_SERVICE_LOCK_STATUSA', '_QUERY_SERVICE_LOCK_STATUSA');
15681: QUERY_SERVICE_LOCK_STATUS', '_QUERY_SERVICE_LOCK_STATUSA');
15682: TQueryServiceLockStatusA', '_QUERY_SERVICE_LOCK_STATUSA');
15683: //TQueryServiceLockStatusW', '_QUERY_SERVICE_LOCK_STATUSW');
15684: TQueryServiceLockStatus', 'TQueryServiceLockStatusA');
15685: _QUERY_SERVICE_CONFIGA', 'record dwServiceType : DWORD; dwStartT'
15686: +'ype : DWORD; dwErrorControl : DWORD; lpBinaryPathName : PChar; lpLoadO'
15687: +'rderGroup : PChar; dwTagId : DWORD; lpDependencies : PChar; lpServ'
15688: +'iceStartTime : PChar; lpDisplayName : PChar; end');
15689: _QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA');
15690: QUERY_SERVICE_CONFIGA', '_QUERY_SERVICE_CONFIGA');
15691: QUERY_SERVICE_CONFIG', '_QUERY_SERVICE_CONFIGA');
15692: TQueryServiceConfigA', '_QUERY_SERVICE_CONFIGA');
15693: TQueryServiceConfig', 'TQueryServiceConfigA');
15694: Function CloseServiceHandle( hSCObject : SC_HANDLE ) : BOOL';
15695: Function ControlService(hService:SC_HANDLE;dwControl DWORD;var lpServiceStatus:TServiceStatus):BOOL;
15696: Function CreateService( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
  dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar;
15697: +'lpdwTagId : DWORD; lpDependencies, lpServiceStartTime, lpPassword : PChar ) : SC_HANDLE );
15698: Function CreateServiceA( hSCManager : SC_HANDLE; lpServiceName, lpDisplayName : PChar; dwDesiredAccess,
  dwServiceType, dwStartType, dwErrorControl : DWORD; lpBinaryPathName, lpLoadOrderGroup : PChar;
15699: +'lpdwTagId : DWORD; lpDependencies, lpServiceStartTime, lpPassword : PChar ) : SC_HANDLE );
15700: Function DeleteService( hService : SC_HANDLE ) : BOOL';
15701: Function EnumDependentServices( hService : SC_HANDLE; dwServiceState : DWORD; var lpServices :
  TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned : DWORD ) : BOOL';
15702: Function EnumServicesStatus( hSCManager : SC_HANDLE; dwServiceType, dwServiceState : DWORD; var
  lpServices : TEnumServiceStatus; cbBufSize : DWORD; var pcbBytesNeeded, lpServicesReturned, lpResumeHandle
  : DWORD):BOOL';
15703: Function GetServiceKeyName(hSCManager:SC_HANDLE;pDisplayName,lpServiceName:PChar;var
  lpcchBuffer:DWORD):BOOL';
15704: Function GetServiceDisplayName(hSCManager:SC_HANDLE;lpServiceName,lpDisplayName:PChar;var
  lpcchBuffer:DWORD):BOOL;
15705: Function LockServiceDatabase( hSCManager : SC_HANDLE ) : SC_LOCK );
15706: Function NotifyBootConfigStatus( BootAcceptable : BOOL ) : BOOL';
15707: Function OpenSCManager( lpMachineName, lpDatabaseName : PChar; dwDesiredAccess : DWORD ) : SC_HANDLE );
15708: Function OpenService(hSCManager:SC_HANDLE;lpServiceName:PChar;dwDesiredAccess:DWORD):SC_HANDLE';
15709: Function QueryServiceLockStatus( hSCManager : SC_HANDLE; var lpLockStatus : TQueryServiceLockStatus;
  cbBufSize : DWORD; var pcbBytesNeeded : DWORD ) : BOOL';
15710: Function QueryServiceStatus(hService:SC_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15711: Function SetServiceStatus(hServiceStatus:SERVICE_STATUS_HANDLE;var lpServiceStatus:TServiceStatus):BOOL;
15712: Function StartService(hService:SC_HANDLE;dwNumServiceArgs:DWORD;var lpServiceArgVectors:PChar):BOOL;
15713: Function UnlockServiceDatabase( ScLock : SC_LOCK ) : BOOL';
15714: end;
15715:
15716: procedure SIRegister_JvPickDate(CL: TPSPascalCompiler);
15717: begin
15718:   Function SelectDate( Sender : TWinControl; var Date : TDateTime; const DlgCaption : TCaption;
  AStartOfWeek : TDayOfWeekName; AWeekends : TDaysOfWeek; AWeekendColor:TColor;
  BtnHints:TStrings;MinDate:TDateTime;MaxDate: TDateTime):Boolean;
15719:   Function SelectDateStr(Sender:TWinControl;var StrDate:string;const
  DlgCaption:TCaption;AStartOfWeek:TDayOfWeekName;AWeekend:TDaysOfWeek;AWeekendClr:TColor;BtnHints:TStrings;MinDate:TDateTime;
15720:   Function PopupDate(var Date:TDateTime; Edit:TWinControl; MinDate:TDateTime; MaxDate:TDateTime):Boolean;
15721:   Function CreatePopupCalendar(AOwner:TComponent;ABiDiMode:TBiDiMode;MinDate:TDateTime;MaxDate:TDateTime):
  TWinControl;
15722:   Procedure SetupPopupCalendar(PopupCalendar:TWinControl;AStartOfWeek:TDayOfWeekName;AWeekends:TDaysOfWeek;
  AWeekendColor:TColor;BtnHints:TStrings;FourDigitYear:Boolean;MinDate:TDateTime;MaxDate:TDateTime);
15723:   Function CreateNotifyThread(const
  FolderName:string;WatchSubtree:Bool;Filter:TFileChangeFilters):TJvNotifyThread;
15724: end;
15725:
15726: procedure SIRegister_JclNTFS2(CL: TPSPascalCompiler);
15727: begin

```

```

15728: CL.AddClassN(CL.FindClass('TOBJECT'),'EJclNtfsError');
15729: CL.AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )');
15730: Function NtfsGetCompression2( const FileName : TFileName; var State : Short ) : Boolean';
15731: Function NtfsGetCompression12( const FileName : TFileName ) : TFileCompressionState';
15732: Function NtfsSetCompression2( const FileName : TFileName; const State : Short ) : Boolean';
15733: Procedure NtfsSetFileCompression2(const FileName : TFileName; const State : TFileCompressionState)';
15734: Procedure NtfsSetDirectoryTreeCompression2(const Directory:string; const State:TFileCompressionState)';
15735: Procedure NtfsSetDefaultFileCompression2(const Directory : string; const State:TFileCompressionState)';
15736: Procedure NtfsSetPathCompression2(const Path:string;const State:TFileCompressionState;Recursive:Bool)';
15737: Function NtfsSetSparse2( const FileName : string ) : Boolean';
15738: Function NtfsZeroDataByHandle2( const Handle : THandle; const First, Last : Int64 ) : Boolean';
15739: Function NtfsZeroDataByName2( const FileName : string; const First, Last : Int64 ) : Boolean';
15740: Function NtfsSparseStreamsSupported2( const Volume : string ) : Boolean';
15741: Function NtfsGetSparse2( const FileName : string ) : Boolean';
15742: Function NtfsDeleteReparsePoint2( const FileName : string; ReparseTag : DWORD ) : Boolean';
15743: Function NtfsSetReparsePoint2( const FileName : string; var ReparseData, Size : Longword ) : Boolean';
15744: Function NtfsGetReparseTag2( const Path : string; var Tag : DWORD ) : Boolean';
15745: Function NtfsReparsePointsSupported2( const Volume : string ) : Boolean';
15746: Function NtfsFileHasReparsePoint2( const Path : string ) : Boolean';
15747: Function NtfsIsFolderMountPoint2( const Path : string ) : Boolean';
15748: Function NtfsMountDeviceAsDrive2( const Device : WideString; Drive : Char ) : Boolean';
15749: Function NtfsMountVolume2( const Volume : WideChar; const MountPoint : WideString ) : Boolean';
15750: CL.AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )');
15751: Function NtfsOpLockAckClosePending2( Handle : THandle; Overlapped : TOverlapped ) : Boolean';
15752: Function NtfsOpLockBreakAckNo22( Handle : THandle; Overlapped : TOverlapped ) : Boolean';
15753: Function NtfsOpLockBreakAcknowledge2( Handle : THandle; Overlapped : TOverlapped ) : Boolean';
15754: Function NtfsOpLockBreakNotify2( Handle : THandle; Overlapped : TOverlapped ) : Boolean';
15755: Function NtfsRequestOpLock2( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped ):Boolean';
15756: Function NtfsCreateJunctionPoint2( const Source, Destination : string ) : Boolean';
15757: Function NtfsDeleteJunctionPoint2( const Source : string ) : Boolean';
15758: Function NtfsGetJunctionPointDestination2( const Source : string; var Destination: string):Boolean;
15759: CL.AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
15760: +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints,siSparseFile)');
15761: CL.AddTypeS('TStreamIds', 'set of TStreamId');
15762: TInternalFindStreamData', 'record FileHandle:THandle; Context: TObject; StreamIds:TStreamIds; end');
15763: CL.AddTypeS('TFindStreamData', 'record Internal : TInternalFindStreamData; At'
15764: +'tributes : DWORD; StreamID : TStreamId; Name: WideString; Size : Int64; end');
15765: Function NtfsFindFirstStream2(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Bool;
15766: Function NtfsFindNextStream2( var Data : TFindStreamData ) : Boolean';
15767: Function NtfsFindStreamClose2( var Data : TFindStreamData ) : Boolean';
15768: Function NtfsCreateHardlink2( const LinkFileName, ExistingFileName : String ) : Boolean';
15769: Function NtfsCreateHardlinkA2( const LinkFileName, ExistingFileName : AnsiString ) : Boolean';
15770: Function NtfsCreateHardlinkW2( const LinkFileName, ExistingFileName : WideString ) : Boolean';
15771: CL.AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end');
15772: Function NtfsGetHardLinkInfo2( const FileName : string; var Info : TNtfsHardLinkInfo ) : Boolean';
15773: Function NtfsFindHardLinks2(const Path:string;const FileIndexHigh,FileIndexLow:Card;const
List:TStrings):Bool
15774: Function NtfsDeleteHardLinks2( const FileName : string ) : Boolean';
15775: FindClass('TOBJECT','EJclFileSummaryError');
15776: TJclFileSummaryAccess', '( fsaRead, fsaWrite, fsaReadWrite )';
15777: TJclFileSummaryShare', '( fssDenyNone, fssDenyRead, fssDenyWrite, fssDenyAll )';
15778: TJclFileSummaryPropSetCallback', 'Function ( const FMTID : TGUID ) : Boolean';
15779: CL.AddClassN(CL.FindClass('TOBJECT','TJclFileSummary'));
15780: SIRegister_TJclFilePropertySet(CL);
15781: //CL.AddTypeS('TJclFilePropertySetClass', 'class of TJclFilePropertySet');
15782: SIRegister_TJclFileSummary(CL);
15783: SIRegister_TJclFileSummaryInformation(CL);
15784: SIRegister_TJclDocSummaryInformation(CL);
15785: SIRegister_TJclMediaFileSummaryInformation(CL);
15786: SIRegister_TJclMSISummaryInformation(CL);
15787: SIRegister_TJclShellSummaryInformation(CL);
15788: SIRegister_TJclStorageSummaryInformation(CL);
15789: SIRegister_TJclImageSummaryInformation(CL);
15790: SIRegister_TJclDisplacedSummaryInformation(CL);
15791: SIRegister_TJclBriefCaseSummaryInformation(CL);
15792: SIRegister_TJclMiscSummaryInformation(CL);
15793: SIRegister_TJclWebViewSummaryInformation(CL);
15794: SIRegister_TJclMusicSummaryInformation(CL);
15795: SIRegister_TJclDRMSummaryInformation(CL);
15796: SIRegister_TJclVideoSummaryInformation(CL);
15797: SIRegister_TJclAudioSummaryInformation(CL);
15798: SIRegister_TJclControlPanelSummaryInformation(CL);
15799: SIRegister_TJclVolumeSummaryInformation(CL);
15800: SIRegister_TJclShareSummaryInformation(CL);
15801: SIRegister_TJclLinkSummaryInformation(CL);
15802: SIRegister_TJclQuerySummaryInformation(CL);
15803: SIRegister_TJclImageInformation(CL);
15804: SIRegister_TJclJpegSummaryInformation(CL);
15805: end;
15806:
15807: procedure SIRegister_Jcl8087(CL: TPPascalCompiler);
15808: begin
15809: AddTypeS('T8087Precision', '( pcSingle, pcReserved, pcDouble, pcExtended )');
15810: T8087Rounding',( rcNearestOrEven, rcDownInfinity, rcUpInfinity, rcChopOrTruncate );
15811: T8087Infinity',( icProjective, icAffine );
15812: T8087Exception',( emInvalidOp,emDenormalizedOperand,emZeroDivide,emOverflow,emUnderflow,emPrecision;
15813: CL.AddTypeS('T8087Exceptions', 'set of T8087Exception');
15814: Function Get8087ControlWord : Word');
15815: Function Get8087Infinity : T8087Infinity');

```

```

15816: Function Get8087Precision : T8087Precision');
15817: Function Get8087Rounding : T8087Rounding');
15818: Function Get8087StatusWord( ClearExceptions : Boolean ) : Word');
15819: Function Set8087Infinity( const Infinity : T8087Infinity ) : T8087Infinity');
15820: Function Set8087Precision( const Precision : T8087Precision ) : T8087Precision');
15821: Function Set8087Rounding( const Rounding : T8087Rounding ) : T8087Rounding');
15822: Function Set8087ControlWord( const Control : Word ) : Word');
15823: Function ClearPending8087Exceptions : T8087Exceptions');
15824: Function GetPending8087Exceptions : T8087Exceptions');
15825: Function GetMasked8087Exceptions : T8087Exceptions');
15826: Function SetMasked8087Exceptions( Exceptions: T8087Exceptions;ClearBefore:Boolean):T8087Exceptions');
15827: Function Mask8087Exceptions( Exceptions:T8087Exceptions ) : T8087Exceptions');
15828: Function Unmask8087Exceptions( Exceptions:T8087Exceptions;ClearBefore : Boolean ) : T8087Exceptions');
15829: end;
15830:
15831: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
15832: begin
15833: Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl );
15834: Procedure BoxMoveAllItems( SrcList, DstList : TWinControl );
15835: Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
Accept:Bool;Sorted:Bool;
15836: Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer );
15837: Procedure BoxMoveSelected( List : TWinControl; Items : TStrings );
15838: Procedure BoxSetItem( List : TWinControl; Index : Integer );
15839: Function BoxGetFirstSelection( List : TWinControl ) : Integer;
15840: Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer ) : Boolean;
15841: end;
15842:
15843: procedure SIRegister_UrlMon(CL: TPSPascalCompiler);
15844: begin
15845: //CL.AddConstantN('SZ_URLCONTEXT','POLEStr').SetString( 'URL Context' );
15846: //CL.AddConstantN('SZ_ASYNC_CALLEE','POLEStr').SetString( 'AsyncCallee' );
15847: CL.AddConstantN('MKSYS_URLMONIKER','LongInt').SetInt( 6 );
15848: type ULONG', 'Cardinal');
15849:     LPCWSTR', 'PChar');
15850: CL.AddTypeS('LPWSTR', 'PChar');
15851: LPSTR', 'PChar');
15852: TBindVerb', 'ULONG');
15853: TBindInfoF', 'ULONG');
15854: TBindF', 'ULONG');
15855: TBindStatus', 'ULONG');
15856: TCIPStatus', 'ULONG');
15857: TBindString', 'ULONG');
15858: TPiFlags', 'ULONG');
15859: TOIBdgFlags', 'ULONG');
15860: TParseAction', 'ULONG');
15861: TPSUAction', 'ULONG');
15862: TQueryOption', 'ULONG');
15863: TPUAF', 'ULONG');
15864: TSZMFlags', 'ULONG');
15865: TUrlZone', 'ULONG');
15866: TUrlTemplate', 'ULONG');
15867: TZAFlags', 'ULONG');
15868: TUrlZoneReg', 'ULONG');
15869: 'URLMON_OPTION_USERAGENT','LongWord').SetUInt( $10000001 );
15870: CL.AddConstantN('URLMON_OPTION_USERAGENT_REFRESH','LongWord').SetUInt( $10000002 );
15871: const 'URLMON_OPTION_URL_ENCODING','LongWord').SetUInt( $10000004 );
15872: const 'URLMON_OPTION_USE_BINDSTRINGCREDS','LongWord').SetUInt( $10000008 );
15873: const 'CF_NULL','LongInt').SetInt( 0 );
15874: const 'CFSTR_MIME_NULL','LongInt').SetInt( 0 );
15875: const 'CFSTR_MIME_TEXT','String').SetString( 'text/plain' );
15876: const 'CFSTR_MIME_RICHTEXT','String').SetString( 'text/richtext' );
15877: const 'CFSTR_MIME_X_BITMAP','String').SetString( 'image/x-bitmap' );
15878: const 'CFSTR_MIME_POSTSCRIPT','String').SetString( 'application/postscript' );
15879: const 'CFSTR_MIME_AIFF','String').SetString( 'audio/aiff' );
15880: const 'CFSTR_MIME_BASICAUDIO','String').SetString( 'audio/basic' );
15881: const 'CFSTR_MIME_WAV','String').SetString( 'audio/wav' );
15882: const 'CFSTR_MIME_X_WAV','String').SetString( 'audio/x-wav' );
15883: const 'CFSTR_MIME_GIF','String').SetString( 'image/gif' );
15884: const 'CFSTR_MIME_PJPEG','String').SetString( 'image/pjpeg' );
15885: const 'CFSTR_MIME_JPEG','String').SetString( 'image/jpeg' );
15886: const 'CFSTR_MIME_TIFF','String').SetString( 'image/tiff' );
15887: const 'CFSTR_MIME_X_PNG','String').SetString( 'image/x-png' );
15888: const 'CFSTR_MIME_BMP','String').SetString( 'image/bmp' );
15889: const 'CFSTR_MIME_X_ART','String').SetString( 'image/x-jg' );
15890: const 'CFSTR_MIME_X_EMF','String').SetString( 'image/x-emf' );
15891: const 'CFSTR_MIME_X_WMF','String').SetString( 'image/x-wmf' );
15892: const 'CFSTR_MIME_AVI','String').SetString( 'video/avi' );
15893: const 'CFSTR_MIME_MPEG','String').SetString( 'video/mpeg' );
15894: const 'CFSTR_MIME_FRACTALS','String').SetString( 'application/fractals' );
15895: const 'CFSTR_MIME_RAWDATA','String').SetString( 'application/octet-stream' );
15896: const 'CFSTR_MIME_RAWDATASTR','String').SetString( 'application/octet-stream' );
15897: const 'CFSTR_MIME_PDF','String').SetString( 'application/pdf' );
15898: const 'CFSTR_MIME_X_AIFF','String').SetString( 'audio/x-aiff' );
15899: const 'CFSTR_MIME_X_REALAUDIO','String').SetString( 'audio/x-pn-realaudio' );
15900: const 'CFSTR_MIME_X_XBML','String').SetString( 'image/xbm' );
15901: const 'CFSTR_MIME_QUICKTIME','String').SetString( 'video/quicktime' );
15902: const 'CFSTR_MIME_X_MSVIDEO','String').SetString( 'video/x-msvideo' );
15903: const 'CFSTR_MIME_X_MSVIDEO','String').SetString( 'video/x-msvideo' );

```

```

15904: const 'CFSTR_MIME_X_SGI_MOVIE','String').SetString('video/x-sgi-movie');
15905: const 'CFSTR_MIME_HTML','String').SetString('text/html');
15906: const 'MK_S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
15907: const 'S_ASYNCHRONOUS','LongWord').SetUInt( $000401E8);
15908: const 'E_PENDING','LongWord').SetUInt( $8000000A);
15909: CL.AddInterface(CL.FindInterface('IUNKNOWN'),IBinding, 'IBinding');
15910: SIRegister_IPersistMoniker(CL);
15911: SIRegister_IBindProtocol(CL);
15912: SIRegister_IBinding(CL);
15913: const 'BINDVERB_GET','LongWord').SetUInt( $00000000);
15914: const 'BINDVERB_POST','LongWord').SetUInt( $00000001);
15915: const 'BINDVERB_PUT','LongWord').SetUInt( $00000002);
15916: const 'BINDVERB_CUSTOM','LongWord').SetUInt( $00000003);
15917: const 'BINDINFO_URLENCODESTGMEDDATA','LongWord').SetUInt( $00000001);
15918: const 'BINDINFO_URLENCODEDEXTRAINFO','LongWord').SetUInt( $00000002);
15919: const 'BINDF_ASYNCHRONOUS','LongWord').SetUInt( $00000001);
15920: const 'BINDF_ASYNCSTORAGE','LongWord').SetUInt( $00000002);
15921: const 'BINDF_NOPROGRESSIVERENDERING','LongWord').SetUInt( $00000004);
15922: const 'BINDF_OFFLINEOPERATION','LongWord').SetUInt( $00000008);
15923: const 'BINDF_GETNEWESTVERSION','LongWord').SetUInt( $00000010);
15924: const 'BINDF_NOWRITECACHE','LongWord').SetUInt( $00000020);
15925: const 'BINDF_NEEDFILE','LongWord').SetUInt( $00000040);
15926: const 'BINDF_PULLDATA','LongWord').SetUInt( $00000080);
15927: const 'BINDF_IGNORESECURITYPROBLEM','LongWord').SetUInt( $00000100);
15928: const 'BINDF_RESYNCHRONIZE','LongWord').SetUInt( $00000200);
15929: const 'BINDF_HYPERLINK','LongWord').SetUInt( $00000400);
15930: const 'BINDF_NO_UI','LongWord').SetUInt( $00000800);
15931: const 'BINDF_SILENTOPERATION','LongWord').SetUInt( $00001000);
15932: const 'BINDF_PRAGMA_NO_CACHE','LongWord').SetUInt( $00002000);
15933: const 'BINDF_FREE_THREADED','LongWord').SetUInt( $00010000);
15934: const 'BINDF_DIRECT_READ','LongWord').SetUInt( $00020000);
15935: const 'BINDF_FORMS_SUBMIT','LongWord').SetUInt( $00040000);
15936: const 'BINDF_GETFROMCACHE_IF_NET_FAIL','LongWord').SetUInt( $00080000);
15937: //const 'BINDF_DONTUSECACHE','').SetString( BINDF_GETNEWESTVERSION);
15938: //const 'BINDF_DONTPUTINCACHE','').SetString( BINDF_NOWRITECACHE);
15939: //const 'BINDF_NOCOPYDATA','').SetString( BINDF_PULLDATA);
15940: const 'BSCF_FIRSTDATANOTIFICATION','LongWord').SetUInt( $00000001);
15941: const 'BSCF_INTERMEDIATEDATANOTIFICATION','LongWord').SetUInt( $00000002);
15942: const 'BSCF_LASTDATANOTIFICATION','LongWord').SetUInt( $00000004);
15943: const 'BSCF_DATAFULLYAVAILABLE','LongWord').SetUInt( $00000008);
15944: const 'BSCF_AVAILABLEDATASIZEUNKNOWN','LongWord').SetUInt( $00000010);
15945: const 'BINDSTATUS_FINDINGRESOURCE','LongInt').SetInt( 1);
15946: const 'BINDSTATUS_CONNECTING','LongInt').SetInt( BINDSTATUS_FINDINGRESOURCE + 1);
15947: const 'BINDSTATUS_REDIRECTING','LongInt').SetInt( BINDSTATUS_CONNECTING + 1);
15948: const 'BINDSTATUS_BEGINDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_REDIRECTING + 1);
15949: const 'BINDSTATUS_DOWNLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADDATA + 1);
15950: const 'BINDSTATUS_ENDDOWNLOADDATA','LongInt').SetInt( BINDSTATUS_DOWNLOADINGDATA + 1);
15951: const 'BINDSTATUS_BEGINDOWNLOADCOMPONENTS','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADDATA + 1);
15952: const 'BINDSTATUS_INSTALLINGCOMPONENTS','LongInt').SetInt( BINDSTATUS_BEGINDOWNLOADCOMPONENTS + 1);
15953: const 'BINDSTATUS_ENDDOWNLOADCOMPONENTS','LongInt').SetInt( BINDSTATUS_INSTALLINGCOMPONENTS + 1);
15954: const 'BINDSTATUS_USINGCACHEDCOPY','LongInt').SetInt( BINDSTATUS_ENDDOWNLOADCOMPONENTS + 1);
15955: const 'BINDSTATUS_SENDINGREQUEST','LongInt').SetInt( BINDSTATUS_USINGCACHEDCOPY + 1);
15956: const 'BINDSTATUS_CLASSIDAVAILABLE','LongInt').SetInt( BINDSTATUS_SENDINGREQUEST + 1);
15957: const 'BINDSTATUS_MIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_CLASSIDAVAILABLE + 1);
15958: const 'BINDSTATUS_CACHEFILENAMEAVAILABLE','LongInt').SetInt( BINDSTATUS_MIMETYPEAVAILABLE + 1);
15959: const 'BINDSTATUS_BEGINSYNCOPERATION','LongInt').SetInt( BINDSTATUS_CACHEFILENAMEAVAILABLE + 1);
15960: const 'BINDSTATUS_ENDSYNCOPERATION','LongInt').SetInt( BINDSTATUS_BEGINSYNCOPERATION + 1);
15961: const 'BINDSTATUS_BEGINUPLOADDATA','LongInt').SetInt( BINDSTATUS_ENDSYNCOPERATION + 1);
15962: const 'BINDSTATUS_UPLOADINGDATA','LongInt').SetInt( BINDSTATUS_BEGINUPLOADDATA + 1);
15963: const 'BINDSTATUS_ENDUPLOADDATA','LongInt').SetInt( BINDSTATUS_UPLOADINGDATA + 1);
15964: const 'BINDSTATUS_PROTOCOLCLASSID','LongInt').SetInt( BINDSTATUS_ENDUPLOADDATA + 1);
15965: const 'BINDSTATUS_ENCODING','LongInt').SetInt( BINDSTATUS_PROTOCOLCLASSID + 1);
15966: const 'BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE','LongInt').SetInt( BINDSTATUS_ENCODING + 1);
15967: const 'BINDSTATUS_CLASSINSTALLLOCATION','LongInt').SetInt( BINDSTATUS_VERIFIEDMIMETYPEAVAILABLE + 1);
15968: const 'BINDSTATUS_DECODING','LongInt').SetInt( BINDSTATUS_CLASSINSTALLLOCATION + 1);
15969: const 'BINDSTATUS_LOADINGMIMEHANDLER','LongInt').SetInt( BINDSTATUS_DECODING + 1);
15970: const 'BINDSTATUS_CONTENTDISPOSITIONATTACH','LongInt').SetInt( BINDSTATUS_LOADINGMIMEHANDLER + 1);
15971: const 'BINDSTATUS_FILTERREPORTMIMETYPE','LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONATTACH + 1);
15972: const 'BINDSTATUS_CLSIDCANINSTANTIATE','LongInt').SetInt( BINDSTATUS_FILTERREPORTMIMETYPE + 1);
15973: const 'BINDSTATUS_IUNKNOWNAVAILABLE','LongInt').SetInt( BINDSTATUS_CLSIDCANINSTANTIATE + 1);
15974: const 'BINDSTATUS_DIRECTBIND','LongInt').SetInt( BINDSTATUS_IUNKNOWNAVAILABLE + 1);
15975: const 'BINDSTATUS_RAWMIMETYPE','LongInt').SetInt( BINDSTATUS_DIRECTBIND + 1);
15976: const 'BINDSTATUS_PROXYDETECTING','LongInt').SetInt( BINDSTATUS_RAWMIMETYPE + 1);
15977: const 'BINDSTATUS_ACCEPTRANGES','LongInt').SetInt( BINDSTATUS_PROXYDETECTING + 1);
15978: const 'BINDSTATUS_COOKIE_SENT','LongInt').SetInt( BINDSTATUS_ACCEPTRANGES + 1);
15979: const 'BINDSTATUS_COMPACT_POLICY RECEIVED','LongInt').SetInt( BINDSTATUS_COOKIE_SENT + 1);
15980: const 'BINDSTATUS_COOKIE_SUPPRESSED','LongInt').SetInt( BINDSTATUS_COMPACT_POLICY RECEIVED + 1);
15981: const 'BINDSTATUS_COOKIE_STATE UNKNOWN','LongInt').SetInt( BINDSTATUS_COOKIE_SUPPRESSED + 1);
15982: const 'BINDSTATUS_COOKIE_STATE ACCEPT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE UNKNOWN + 1);
15983: const 'BINDSTATUS_COOKIE_STATE REJECT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE ACCEPT + 1);
15984: const 'BINDSTATUS_COOKIE_STATE PROMPT','LongInt').SetInt( BINDSTATUS_COOKIE_STATE REJECT + 1);
15985: const 'BINDSTATUS_COOKIE_STATE LEASH','LongInt').SetInt( BINDSTATUS_COOKIE_STATE PROMPT + 1);
15986: const 'BINDSTATUS_COOKIE_STATE DOWNGRADE','LongInt').SetInt( BINDSTATUS_COOKIE_STATE LEASH + 1);
15987: const 'BINDSTATUS_POLICY_HREF','LongInt').SetInt( BINDSTATUS_COOKIE_STATE DOWNGRADE + 1);
15988: const 'BINDSTATUS_P3P_HEADER','LongInt').SetInt( BINDSTATUS_POLICY_HREF + 1);
15989: const 'BINDSTATUS_SESSION_COOKIE RECEIVED','LongInt').SetInt( BINDSTATUS_P3P_HEADER + 1);
15990: const 'BINDSTATUS_PERSISTENT_COOKIE RECEIVED','LongInt').SetInt( BINDSTATUS_SESSION_COOKIE RECEIVED + 1);
15991: const 'BINDSTATUS_SESSION_COOKIES_ALLOWED','LongInt').SetInt( BINDSTATUS_PERSISTENT_COOKIE RECEIVED + 1);
15992: const 'BINDSTATUS_CACHECONTROL','LongInt').SetInt( BINDSTATUS_SESSION_COOKIES_ALLOWED + 1);

```

```

15993: const 'BINDSTATUS_CONTENTDISPOSITIONFILENAME','LongInt').SetInt( BINDSTATUS_CACHECONTROL + 1);
15994: const 'BINDSTATUS_MIMETEXTPLAINMISMATCH','LongInt').SetInt( BINDSTATUS_CONTENTDISPOSITIONFILENAME + 1);
15995: const 'BINDSTATUS_PUBLISHERAVAILABLE','LongInt').SetInt( BINDSTATUS_MIMETEXTPLAINMISMATCH + 1);
15996: const 'BINDSTATUS_DISPLAYNAMEAVAILABLE','LongInt').SetInt( BINDSTATUS_PUBLISHERAVAILABLE + 1);
15997: // PBindInfo', '^TBindInfo // will not work');
15998: {_tagBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; stg'
15999: +'medData : TStgMedium; grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVe'
16000: +'rb : LPWSTR; cbstgmedData : DWORD; dwOptions : DWORD; dwOptionsFlags : DWO'
16001: +'RD; dwCodePage : DWORD; securityAttributes : TSecurityAttributes; iid : TG'
16002: +'UID; pUnk : IUnknown; dwReserved : DWORD; end');
16003: TBindInfo', '_tagBINDINFO');
16004: BINDINFO', '_tagBINDINFO');
16005: _REMSecurity_ATTRIBUTES', 'record nLength : DWORD; lpSecurityDes'
16006: +'criptor : DWORD; bInheritHandle : BOOL; end');
16007: TRemSecurityAttributes', '_REMSecurity_ATTRIBUTES');
16008: REMSECURITY_ATTRIBUTES', '_REMSecurity_ATTRIBUTES');
16009: //PREmBindInfo', '^TRemBindInfo // will not work');
16010: {_tagRemBINDINFO', 'record cbSize : ULONG; szExtraInfo : LPWSTR; '
16011: +'grfBindInfoF : DWORD; dwBindVerb : DWORD; szCustomVerb : LPWSTR; cbstgmedD'
16012: +'ata : DWORD; dwOptions : DWORD; dwOptionsFlags : DWORD; dwCodePage : DWORD'
16013: +'; securityAttributes : TRemSecurityAttributes; iid : TGUID; pUnk : IUnknown'
16014: +'n; dwReserved : DWORD; end');
16015: TRemBindInfo', '_tagRemBINDINFO');
16016: RemBINDINFO', '_tagRemBINDINFO');
16017: //PREmFormatEtc', '^TRemFormatEtc // will not work');
16018: tagRemFORMATETC', 'record cfFormat:DWORD; ptd:DWORD; dwAspect:DWORD;lindex:Longint;tmmed:DWORD; end');
16019: TRemFormatEtc', 'tagRemFORMATETC');
16020: RemFORMATETC', 'tagRemFORMATETC');
16021: SIRegister_IBindStatusCallback(CL);
16022: SIRegister_IAuthenticate(CL);
16023: SIRegister_IHttpNegotiate(CL);
16024: SIRegister_IWindowForBindingUI(CL);
16025: const 'CIP_DISK_FULL','LongInt').SetInt( 0);
16026: const 'CIP_ACCESS_DENIED','LongInt').SetInt( CIP_DISK_FULL + 1);
16027: const 'CIP_NEWER_VERSION_EXISTS','LongInt').SetInt( CIP_ACCESS_DENIED + 1);
16028: const 'CIP_Older_VERSION_EXISTS','LongInt').SetInt( CIP_NEWER_VERSION_EXISTS + 1);
16029: const 'CIP_NAME_CONFLICT','LongInt').SetInt( CIP_Older_VERSION_EXISTS + 1);
16030: const 'CIP_TRUST_VERIFICATION_COMPONENT_MISSING','LongInt').SetInt( CIP_NAME_CONFLICT + 1);
16031: const 'CIP_EXE_SELF_REGISTRATION_TIMEOUT','LongInt').SetInt( CIP_TRUST_VERIFICATION_COMPONENT_MISSING + 1);
16032: const 'CIP_UNSAFE_TO_ABORT','LongInt').SetInt( CIP_EXE_SELF_REGISTRATION_TIMEOUT + 1);
16033: const 'CIP_NEED_REBOOT','LongInt').SetInt( CIP_UNSAFE_TO_ABORT + 1);
16034: const 'CIP_NEED_REBOOT_UI_PERMISSION','LongInt').SetInt( CIP_NEED_REBOOT + 1);
16035: SIRegister_ICodeInstall(CL);
16036: SIRegister_IWInetInfo(CL);
16037: const 'WININETINFO_OPTION_LOCK_HANDLE','LongInt').SetInt( 65534);
16038: SIRegister_IHttpSecurity(CL);
16039: SIRegister_IWInetHttpInfo(CL);
16040: SIRegister_IBindHost(CL);
16041: const 'URLOSTRM_USECACHEDCOPY_ONLY','LongWord').SetUInt( $00000001);
16042: const 'URLOSTRM_USECACHEDCOPY','LongWord').SetUInt( $00000002);
16043: const 'URLOSTRM_GETNEWESTVERSION','LongWord').SetUInt( $00000003);
16044: Function URLOpenStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; p4 : IBindStatusCallback ) : HResult';
16045: Function URLOpenPullStream( p1 : IUnknown; p2 : PChar; p3 : DWORD; BSC : IBindStatusCallback ) : HResult');
16046: Function URLDownloadToFile( Caller : IUnknown; URL : PChar; FileName : PChar; Reserved : DWORD; StatusCB
: IBindStatusCallback ) : HResult');
16047: Function URLDownloadToCacheFile( p1 : IUnknown; p2 : PChar; p3 : PChar; p4 : DWORD; p5 : DWORD; p6 :
IBindStatusCallback ) : HResult');
16048: Function URLOpenBlockingStream(p1:IUnknown;p2:PChar;out
p3:IStream;p4:DWORD;p5:IBindStatusCallback:HResult');
16049: Function HlinkGoBack( unk : IUnknown ) : HResult';
16050: Function HlinkGoForward( unk : IUnknown ) : HResult';
16051: Function HlinkNavigateString( unk : IUnknown; szTarget : LPCWSTR ) : HResult';
16052: // Function HlinkNavigateMoniker( unk : IUnknown; mkTarget : IMoniker ) : HResult');
16053: SIRegister_IInternet(CL);
16054: const 'BINDSTRING_HEADERS','LongInt').SetInt( 1);
16055: const 'BINDSTRING_ACCEPT_MIMES','LongInt').SetInt( BINDSTRING_HEADERS + 1);
16056: const 'BINDSTRING_EXTRA_URL','LongInt').SetInt( BINDSTRING_ACCEPT_MIMES + 1);
16057: const 'BINDSTRING_LANGUAGE','LongInt').SetInt( BINDSTRING_EXTRA_URL + 1);
16058: const 'BINDSTRING_USERNAME','LongInt').SetInt( BINDSTRING_LANGUAGE + 1);
16059: const 'BINDSTRING_PASSWORD','LongInt').SetInt( BINDSTRING_USERNAME + 1);
16060: const 'BINDSTRING_UA_PIXELS','LongInt').SetInt( BINDSTRING_PASSWORD + 1);
16061: const 'BINDSTRING_UA_COLOR','LongInt').SetInt( BINDSTRING_UA_PIXELS + 1);
16062: const 'BINDSTRING_OS','LongInt').SetInt( BINDSTRING_UA_COLOR + 1);
16063: const 'BINDSTRING_USER_AGENT','LongInt').SetInt( BINDSTRING_OS + 1);
16064: const 'BINDSTRING_ACCEPT_ENCODINGS','LongInt').SetInt( BINDSTRING_USER_AGENT + 1);
16065: const 'BINDSTRING_POST_COOKIE','LongInt').SetInt( BINDSTRING_ACCEPT_ENCODINGS + 1);
16066: const 'BINDSTRING_POST_DATA_MIME','LongInt').SetInt( BINDSTRING_POST_COOKIE + 1);
16067: const 'BINDSTRING_URL','LongInt').SetInt( BINDSTRING_POST_DATA_MIME + 1);
16068: //POLEStrArray', '^TOLESTRArray // will not work';
16069: SIRegister_IInternetBindInfo(CL);
16070: const 'PI_PARSE_URL','LongWord').SetUInt( $00000001);
16071: const 'PI_FILTER_MODE','LongWord').SetUInt( $00000002);
16072: const 'PI_FORCE_ASYNC','LongWord').SetUInt( $00000004);
16073: const 'PI_USE_WORKERTHREAD','LongWord').SetUInt( $00000008);
16074: const 'PI_MIMEVERIFICATION','LongWord').SetUInt( $00000010);
16075: const 'PI_CLSIDLOOKUP','LongWord').SetUInt( $00000020);
16076: const 'PI_DATAPROGRESS','LongWord').SetUInt( $00000040);
16077: const 'PI_SYNCHRONOUS','LongWord').SetUInt( $00000080);
16078: const 'PI_APARTMENTTHREADED','LongWord').SetUInt( $00000100);

```

```

16079: const 'PI_CLASSINSTALL','LongWord').SetUInt( $00000200);
16080: const 'PD_FORCE_SWITCH','LongWord').SetUInt( $00010000);
16081: //const 'PI_DOCFILECLSIDLOOKUP','').SetString( PI_CLSIDLOOKUP);
16082: //PProtocolData', '^TProtocolData // will not work');
16083: _tagPROTOCOLDATA', record grfFlags:DWORD; dwState:DWORD; pData : TObject; cbData : ULONG; end');
16084: TProtocolData', '_tagPROTOCOLDATA');
16085: PROTOCOLDATA', '_tagPROTOCOLDATA');
16086: CL.AddInterface(CL.FindInterface('IUNKNOWN'),IIInternetProtocolSink, 'IIInternetProtocolSink');
16087: SIRegister_IInternetProtocolRoot(CL);
16088: SIRegister_IInternetProtocol(CL);
16089: SIRegister_IInternetProtocolSink(CL);
16090: const 'OIBDG_APARTMENTTHREADED','LongWord').SetUInt( $00000100);
16091: SIRegister_IInternetSession(CL);
16092: SIRegister_IInternetThreadSwitch(CL);
16093: SIRegister_IInternetPriority(CL);
16094: const 'PARSE_CANONICALIZE','LongInt').SetInt( 1);
16095: const 'PARSE_FRIENDLY','LongInt').SetInt( PARSE_CANONICALIZE + 1);
16096: const 'PARSE_SECURITY_URL','LongInt').SetInt( PARSE_FRIENDLY + 1);
16097: const 'PARSE_ROOTDOCUMENT','LongInt').SetInt( PARSE_SECURITY_URL + 1);
16098: const 'PARSE_DOCUMENT','LongInt').SetInt( PARSE_ROOTDOCUMENT + 1);
16099: const 'PARSE_ANCHOR','LongInt').SetInt( PARSE_DOCUMENT + 1);
16100: const 'PARSE_ENCODE','LongInt').SetInt( PARSE_ANCHOR + 1);
16101: const 'PARSE_DECODE','LongInt').SetInt( PARSE_ENCODE + 1);
16102: const 'PARSE_PATH_FROM_URL','LongInt').SetInt( PARSE_DECODE + 1);
16103: const 'PARSE_URL_FROM_PATH','LongInt').SetInt( PARSE_PATH_FROM_URL + 1);
16104: const 'PARSE_MIME','LongInt').SetInt( PARSE_URL_FROM_PATH + 1);
16105: const 'PARSE_SERVER','LongInt').SetInt( PARSE_MIME + 1);
16106: const 'PARSE_SCHEMA','LongInt').SetInt( PARSE_SERVER + 1);
16107: const 'PARSE_SITE','LongInt').SetInt( PARSE_SCHEMA + 1);
16108: const 'PARSE_DOMAIN','LongInt').SetInt( PARSE_SITE + 1);
16109: const 'PARSE_LOCATION','LongInt').SetInt( PARSE_DOMAIN + 1);
16110: const 'PARSE_SECURITY_DOMAIN','LongInt').SetInt( PARSE_LOCATION + 1);
16111: const 'PSU_DEFAULT','LongInt').SetInt( 1);
16112: const 'PSU_SECURITY_URL_ONLY','LongInt').SetInt( PSU_DEFAULT + 1);
16113: const 'QUERY_EXPIRATION_DATE','LongInt').SetInt( 1);
16114: const 'QUERY_TIME_OF_LAST_CHANGE','LongInt').SetInt( QUERY_EXPIRATION_DATE + 1);
16115: const 'QUERY_CONTENT_ENCODING','LongInt').SetInt( QUERY_TIME_OF_LAST_CHANGE + 1);
16116: const 'QUERY_CONTENT_TYPE','LongInt').SetInt( QUERY_CONTENT_ENCODING + 1);
16117: const 'QUERY_REFRESH','LongInt').SetInt( QUERY_CONTENT_TYPE + 1);
16118: const 'QUERY_RECOMBINE','LongInt').SetInt( QUERY_REFRESH + 1);
16119: const 'QUERY_CAN_NAVIGATE','LongInt').SetInt( QUERY_RECOMBINE + 1);
16120: const 'QUERYUSES_NETWORK','LongInt').SetInt( QUERY_CAN_NAVIGATE + 1);
16121: const 'QUERY_IS_CACHED','LongInt').SetInt( QUERYUSES_NETWORK + 1);
16122: const 'QUERY_IS_INSTALLEDENTRY','LongInt').SetInt( QUERY_IS_CACHED + 1);
16123: const 'QUERY_IS_CACHED_OR_MAPPED','LongInt').SetInt( QUERY_IS_INSTALLEDENTRY + 1);
16124: const 'QUERYUSES_CACHE','LongInt').SetInt( QUERY_IS_CACHED_OR_MAPPED + 1);
16125: const 'QUERY_IS_SECURE','LongInt').SetInt( QUERYUSES_CACHE + 1);
16126: const 'QUERY_IS_SAFE','LongInt').SetInt( QUERY_IS_SECURE + 1);
16127: SIRegister_IInternetProtocolInfo(CL);
16128: IOInet', 'IInternet');
16129: IOInetBindInfo', 'IInternetBindInfo');
16130: IOInetProtocolRoot', 'IInternetProtocolRoot');
16131: IOInetProtocol', 'IInternetProtocol');
16132: IOInetProtocolSink', 'IInternetProtocolSink');
16133: IOInetProtocolInfo', 'IInternetProtocolInfo');
16134: IOInetSession', 'IInternetSession');
16135: IOInetPriority', 'IInternetPriority');
16136: IOInetThreadSwitch', 'IInternetThreadSwitch');
16137: Function CoInternetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult : LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16138: Function CoInternetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult : LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16139: Function CoInternetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : HResult';
16140: Function CoInternetGetProtocolFlags( pwzUrl:LPCWSTR;var dwFlags DWORD; dwReserved:DWORD):HResult');
16141: Function CoInternetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD; pvBuffer : TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult');
16142: Function CoInternetGetSession(dwSessionMode:DWORD; var piInternetSession: IInternetSes;dwReserved:DWORD):HResult;
16143: Function CoInternetGetSecurityUrl(pwzUrl:LPCWSTR;var pwzSecUrl:LPWSTR;psuAct:TProtocolHandler;dwReserv:DWORD):HResult;
16144: Function OInetParseUrl( pwzUrl : LPCWSTR; ParseAction : TParseAction; dwFlags : DWORD; pszResult : LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16145: Function OInetCombineUrl( pwzBaseUrl, pwzRelativeUrl : LPCWSTR; dwCombineFlags : DWORD; pszResult : LPWSTR; cchResult : DWORD; var pcchResult : DWORD; dwReserved : DWORD) : HResult');
16146: Function OInetCompareUrl( pwzUrl1, pwzUrl2 : LPCWSTR; dwFlags : DWORD) : Hresult';
16147: Function OInetGetProtocolFlags( pwzUrl : LPCWSTR; var dwFlags : DWORD; dwReserved : DWORD) : HResult');
16148: Function OInetQueryInfo( pwzUrl : LPCWSTR; QueryOptions : TQueryOption; dwQueryFlags : DWORD; pvBuffer : TObject; cbBuffer : DWORD; var pcbBuffer : DWORD; dwReserved : DWORD) : HResult');
16149: Function OInetGetSession(dwSessionMode:DWORD; var piInternetSession:IInternetSession;dwReserved:DWORD):HResult;
16150: //Function CopyBindInfo( const cbisrc : TBindInfo; var bidest : TBindInfo) : HResult';
16151: //Procedure ReleaseBindInfo( const bindinfo : TBindInfo);
16152: // const 'INET_E_USE_DEFAULT_PROTOCOLHANDLER','LongWord').SetUInt( HResult ( $800C0011 ));
16153: // const 'INET_E_USE_DEFAULT_SETTING','LongWord').SetUInt( HResult ( $800C0012 ));
16154: //const 'INET_E_DEFAULT_ACTION','LongWord').SetUInt( HResult ( $800C0011 ));
16155: //const 'INET_E_QUERYOPTION_UNKNOWN','LongWord').SetUInt( HResult ( $800C0013 ));
16156: //const 'INET_E_REDIRECTING','LongWord').SetUInt( HResult ( $800C0014 ));
16157: const 'PROTOCOLFLAG_NO_PICS_CHECK','LongWord').SetUInt( $00000001);
16158: SIRegister_IInternetSecurityMgrSite(CL);

```

```

16159: const 'MUTZ_NOSAVEDFILECHECK','LongWord').SetUInt( $00000001);
16160: const 'MUTZ_ISFILE','LongWord').SetUInt( $00000002);
16161: const 'MUTZ_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16162: const 'MUTZ_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16163: const 'MUTZ_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16164: const 'MAX_SIZE_SECURITY_ID','LongInt').SetInt( 512);
16165: const 'PUAF_DEFAULT','LongWord').SetUInt( $00000000);
16166: const 'PUAF_NOUI','LongWord').SetUInt( $00000001);
16167: const 'PUAF_ISFILE','LongWord').SetUInt( $00000002);
16168: const 'PUAF_WARN_IF_DENIED','LongWord').SetUInt( $00000004);
16169: const 'PUAF_FORCEUI_FOREGROUND','LongWord').SetUInt( $00000008);
16170: const 'PUAF_CHECK_TIFS','LongWord').SetUInt( $00000010);
16171: const 'PUAF_DONTCHECKBOXINDIALOG','LongWord').SetUInt( $00000020);
16172: const 'PUAF_TRUSTED','LongWord').SetUInt( $00000040);
16173: const 'PUAF_ACCEPT_WILDCARD_SCHEME','LongWord').SetUInt( $00000080);
16174: const 'PUAF_ENFORCERESTRICTED','LongWord').SetUInt( $00000100);
16175: const 'PUAF_NOSAVEDFILECHECK','LongWord').SetUInt( $00000200);
16176: const 'PUAF_REQUIRESAVEDFILECHECK','LongWord').SetUInt( $00000400);
16177: const 'PUAF_LMZ_UNLOCKED','LongWord').SetUInt( $00010000);
16178: const 'PUAF_LMZ_LOCKED','LongWord').SetUInt( $00020000);
16179: const 'PUAF_DEFAULTZONEPOL','LongWord').SetUInt( $00040000);
16180: const 'PUAF_NPL_USE_LOCKED_IF_RESTRICTED','LongWord').SetUInt( $00080000);
16181: const 'PUAF_NOUIIFLOCKED','LongWord').SetUInt( $00100000);
16182: const 'PUAFOUT_DEFAULT','LongWord').SetUInt( $0);
16183: const 'PUAFOUT_ISLOCKZONEPOLICY','LongWord').SetUInt( $1);
16184: const 'SZM_CREATE','LongWord').SetUInt( $00000000);
16185: const 'SZM_DELETE','LongWord').SetUInt( $00000001);
16186: SIRegister_IInternetSecurityManager(CL);
16187: SIRegister_IInternetHostSecurityManager(CL);
16188: SIRegister_IInternetSecurityManagerEx(CL);
16189: const 'URLACTION_MIN','LongWord').SetUInt( $00001000);
16190: const 'URLACTION_DOWNLOAD_MIN','LongWord').SetUInt( $00001000);
16191: const 'URLACTION_DOWNLOAD_SIGNED_ACTIVEX','LongWord').SetUInt( $00001001);
16192: const 'URLACTION_DOWNLOAD_UNSIGNED_ACTIVEX','LongWord').SetUInt( $00001004);
16193: const 'URLACTION_DOWNLOAD_CURR_MAX','LongWord').SetUInt( $00001004);
16194: const 'URLACTION_DOWNLOAD_MAX','LongWord').SetUInt( $000011FF);
16195: const 'URLACTION_ACTIVEX_MIN','LongWord').SetUInt( $00001200);
16196: const 'URLACTION_ACTIVEX_RUN','LongWord').SetUInt( $00001200);
16197: const 'URLACTION_ACTIVEX_OVERRIDE_OBJECT_SAFETY','LongWord').SetUInt( $00001201);
16198: const 'URLACTION_ACTIVEX_OVERRIDE_DATA_SAFETY','LongWord').SetUInt( $00001202);
16199: const 'URLACTION_ACTIVEX_OVERRIDE_SCRIPT_SAFETY','LongWord').SetUInt( $00001203);
16200: const 'URLACTION_SCRIPT_OVERRIDE_SAFETY','LongWord').SetUInt( $00001401);
16201: const 'URLACTION_ACTIVEX_CONFIRM_NOOBJECTSAFETY','LongWord').SetUInt( $00001204);
16202: const 'URLACTION_ACTIVEX_TREATASUNTRUSTED','LongWord').SetUInt( $00001205);
16203: const 'URLACTION_ACTIVEX_NO_WEBOC_SCRIPT','LongWord').SetUInt( $00001206);
16204: const 'URLACTION_ACTIVEX_CURR_MAX','LongWord').SetUInt( $00001206);
16205: const 'URLACTION_ACTIVEX_MAX','LongWord').SetUInt( $000013FF);
16206: const 'URLACTION_SCRIPT_MIN','LongWord').SetUInt( $00001400);
16207: const 'URLACTION_SCRIPT_RUN','LongWord').SetUInt( $00001400);
16208: const 'URLACTION_SCRIPT_JAVA_USE','LongWord').SetUInt( $00001402);
16209: const 'URLACTION_SCRIPT_SAFE_ACTIVEX','LongWord').SetUInt( $00001405);
16210: const 'URLACTION_SCRIPT_CURR_MAX','LongWord').SetUInt( $00001405);
16211: const 'URLACTION_SCRIPT_MAX','LongWord').SetUInt( $000015FF);
16212: const 'URLACTION_HTML_MIN','LongWord').SetUInt( $00001600);
16213: const 'URLACTION_HTML_SUBMIT_FORMS','LongWord').SetUInt( $00001601);
16214: const 'URLACTION_HTML_SUBMIT_FORMS_FROM','LongWord').SetUInt( $00001602);
16215: const 'URLACTION_HTML_SUBMIT_FORMS_TO','LongWord').SetUInt( $00001603);
16216: const 'URLACTION_HTML_FONT_DOWNLOAD','LongWord').SetUInt( $00001604);
16217: const 'URLACTION_HTML_JAVA_RUN','LongWord').SetUInt( $00001605);
16218: const 'URLACTION_HTML_CURR_MAX','LongWord').SetUInt( $00001605);
16219: const 'URLACTION_HTML_MAX','LongWord').SetUInt( $000017FF);
16220: const 'URLACTION_SHELL_MIN','LongWord').SetUInt( $00001800);
16221: const 'URLACTION_SHELL_INSTALL_DTITEMS','LongWord').SetUInt( $00001800);
16222: const 'URLACTION_SHELL_MOVE_OR_COPY','LongWord').SetUInt( $00001802);
16223: const 'URLACTION_SHELL_FILE_DOWNLOAD','LongWord').SetUInt( $00001803);
16224: const 'URLACTION_SHELL_VERB','LongWord').SetUInt( $00001804);
16225: const 'URLACTION_SHELL_WEBVIEW_VERB','LongWord').SetUInt( $00001805);
16226: const 'URLACTION_SHELL_SHELLEXECUTE','LongWord').SetUInt( $00001806);
16227: const 'URLACTION_SHELL_EXECUTE_HIGHRISK','LongWord').SetUInt( $00001806);
16228: const 'URLACTION_SHELL_EXECUTE_MODRISK','LongWord').SetUInt( $00001807);
16229: const 'URLACTION_SHELL_EXECUTE_LOWRISK','LongWord').SetUInt( $00001808);
16230: const 'URLACTION_SHELL_POPUPMGR','LongWord').SetUInt( $00001809);
16231: const 'URLACTION_SHELL_CURR_MAX','LongWord').SetUInt( $00001809);
16232: const 'URLACTION_SHELL_MAX','LongWord').SetUInt( $000019FF);
16233: const 'URLACTION_NETWORK_MIN','LongWord').SetUInt( $00001A00);
16234: const 'URLACTION_CREDENTIALS_USE','LongWord').SetUInt( $00001A00);
16235: const 'URLPOLICY_CREDENTIALS_SILENT_LOGON_OK','LongWord').SetUInt( $00000000);
16236: const 'URLPOLICY_CREDENTIALS_MUST_PROMPT_USER','LongWord').SetUInt( $00010000);
16237: const 'URLPOLICY_CREDENTIALS_CONDITIONAL_PROMPT','LongWord').SetUInt( $00020000);
16238: const 'URLPOLICY_CREDENTIALS_ANONYMOUS_ONLY','LongWord').SetUInt( $00030000);
16239: const 'URLACTION_AUTHENTICATE_CLIENT','LongWord').SetUInt( $00001A01);
16240: const 'URLPOLICY_AUTHENTICATE_CLEARTEXT_OK','LongWord').SetUInt( $00000000);
16241: const 'URLPOLICY_AUTHENTICATE_CHALLENGE_RESPONSE','LongWord').SetUInt( $00010000);
16242: const 'URLPOLICY_AUTHENTICATE_MUTUAL_ONLY','LongWord').SetUInt( $00030000);
16243: const 'URLACTION_NETWORK_CURR_MAX','LongWord').SetUInt( $00001A01);
16244: const 'URLACTION_NETWORK_MAX','LongWord').SetUInt( $00001BFF);
16245: const 'URLACTION_JAVA_MIN','LongWord').SetUInt( $00001C00);
16246: const 'URLACTION_JAVA_PERMISSIONS','LongWord').SetUInt( $00001C00);
16247: const 'URLPOLICY_JAVA_PROHIBIT','LongWord').SetUInt( $00000000);

```

```

16248: const 'URLPOLICY_JAVA_HIGH','LongWord').SetUInt( $00010000);
16249: const 'URLPOLICY_JAVA_MEDIUM','LongWord').SetUInt( $00020000);
16250: const 'URLPOLICY_JAVA_LOW','LongWord').SetUInt( $00030000);
16251: const 'URLPOLICY_JAVA_CUSTOM','LongWord').SetUInt( $00800000);
16252: const 'URLACTION_JAVA_CURR_MAX','LongWord').SetUInt( $00001C00);
16253: const 'URLACTION_JAVA_MAX','LongWord').SetUInt( $00001CFF);
16254: const 'URLACTION_INFODELIVERY_MIN','LongWord').SetUInt( $00001D00);
16255: const 'URLACTION_INFODELIVERY_NO_ADDING_CHANNELS','LongWord').SetUInt( $00001D00);
16256: const 'URLACTION_INFODELIVERY_NO_EDITING_CHANNELS','LongWord').SetUInt( $00001D01);
16257: const 'URLACTION_INFODELIVERY_NO_Removing_CHANNELS','LongWord').SetUInt( $00001D02);
16258: const 'URLACTION_INFODELIVERY_NO_ADDING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D03);
16259: const 'URLACTION_INFODELIVERY_NO_EDITING_SUBSCRIPTIONS','LongWord').SetUInt( $00001D04);
16260: const 'URLACTION_INFODELIVERY_NO_Removing_SUBSCRIPTIONS','LongWord').SetUInt( $00001D05);
16261: const 'URLACTION_INFODELIVERY_NO_CHANNEL_LOGGING','LongWord').SetUInt( $00001D06);
16262: const 'URLACTION_INFODELIVERY_Curr_MAX','LongWord').SetUInt( $00001D06);
16263: const 'URLACTION_INFODELIVERY_MAX','LongWord').SetUInt( $00001Dff);
16264: const 'URLACTION_CHANNEL_SOFTDIST_MIN','LongWord').SetUInt( $00001E00);
16265: const 'URLACTION_CHANNEL_SOFTDIST_PERMISSIONS','LongWord').SetUInt( $00001E05);
16266: const 'URLPOLICY_CHANNEL_SOFTDIST_PROHIBIT','LongWord').SetUInt( $00010000);
16267: const 'URLPOLICY_CHANNEL_SOFTDIST_PRECACHE','LongWord').SetUInt( $00020000);
16268: const 'URLPOLICY_CHANNEL_SOFTDIST_AUTOINSTALL','LongWord').SetUInt( $00030000);
16269: const 'URLACTION_CHANNEL_SOFTDIST_MAX','LongWord').SetUInt( $00001EFF);
16270: const 'URLACTION_BEHAVIOR_MIN','LongWord').SetUInt( $00002000);
16271: const 'URLACTION_BEHAVIOR_RUN','LongWord').SetUInt( $00002000);
16272: const 'URLPOLICY_BEHAVIOR_CHECK_LIST','LongWord').SetUInt( $00010000);
16273: const 'URLACTION_FEATURE_MIN','LongWord').SetUInt( $00002100);
16274: const 'URLACTION_FEATURE_MIME_SNIFFING','LongWord').SetUInt( $00002100);
16275: const 'URLACTION_FEATURE_ZONE_ELEVATION','LongWord').SetUInt( $00002101);
16276: const 'URLACTION_FEATURE_WINDOW_RESTRICTIONS','LongWord').SetUInt( $00002102);
16277: const 'URLACTION_AUTOMATIC_DOWNLOAD_UI_MIN','LongWord').SetUInt( $00002200);
16278: const 'URLACTION_AUTOMATIC_DOWNLOAD_UI','LongWord').SetUInt( $00002200);
16279: const 'URLACTION_AUTOMATIC_ACTIVEX_UI','LongWord').SetUInt( $00002201);
16280: const 'URLACTION_ALLOW_RESTRICTEDPROTOCOLS','LongWord').SetUInt( $00002300);
16281: const 'URLPOLICY_ALLOW','LongWord').SetUInt( $00);
16282: const 'URLPOLICY_QUERY','LongWord').SetUInt( $01);
16283: const 'URLPOLICY_DISALLOW','LongWord').SetUInt( $03);
16284: const 'URLPOLICY_NOTIFY_ON_ALLOW','LongWord').SetUInt( $10);
16285: const 'URLPOLICY_NOTIFY_ON_DISALLOW','LongWord').SetUInt( $20);
16286: const 'URLPOLICY_LOG_ON_ALLOW','LongWord').SetUInt( $40);
16287: const 'URLPOLICY_LOG_ON_DISALLOW','LongWord').SetUInt( $80);
16288: const 'URLPOLICY_MASK_PERMISSIONS','LongWord').SetUInt( $0f);
16289: Function GetUrlPolicyPermissions( dw : DWORD ) : DWORD';
16290: Function SetUrlPolicyPermissions( dw, dw2 : DWORD ) : DWORD';
16291: const 'URLZONE_PREDEFINED_MIN','LongInt').SetInt( 0);
16292: const 'URLZONE_LOCAL_MACHINE','LongInt').SetInt( 0);
16293: const 'URLZONE_INTRANET','LongInt').SetInt( URLZONE_LOCAL_MACHINE + 1);
16294: const 'URLZONE_TRUSTED','LongInt').SetInt( URLZONE_INTRANET + 1);
16295: const 'URLZONE_INTERNET','LongInt').SetInt( URLZONE_TRUSTED + 1);
16296: const 'URLZONE_UNTRUSTED','LongInt').SetInt( URLZONE_INTERNET + 1);
16297: const 'URLZONE_PREDEFINED_MAX','LongInt').SetInt( 999);
16298: const 'URLZONE_USER_MIN','LongInt').SetInt( 1000);
16299: const 'URLZONE_USER_MAX','LongInt').SetInt( 10000);
16300: const 'URLTEMPLATE_CUSTOM','LongWord').SetUInt( $00000000);
16301: const 'URLTEMPLATE_PREDEFINED_MIN','LongWord').SetUInt( $00010000);
16302: const 'URLTEMPLATE_LOW','LongWord').SetUInt( $00010000);
16303: const 'URLTEMPLATE_MEDIUM','LongWord').SetUInt( $00011000);
16304: const 'URLTEMPLATE_HIGH','LongWord').SetUInt( $00012000);
16305: const 'URLTEMPLATE_PREDEFINED_MAX','LongWord').SetUInt( $00020000);
16306: const 'MAX_ZONE_PATH','LongInt').SetInt( 260);
16307: const 'MAX_ZONE_DESCRIPTION','LongInt').SetInt( 200);
16308: const 'ZAFLAGS_CUSTOM_EDIT','LongWord').SetUInt( $00000001);
16309: const 'ZAFLAGS_ADD_SITES','LongWord').SetUInt( $00000002);
16310: const 'ZAFLAGS_REQUIRE_VERIFICATION','LongWord').SetUInt( $00000004);
16311: const 'ZAFLAGS_INCLUDE_PROXY_OVERRIDE','LongWord').SetUInt( $00000008);
16312: const 'ZAFLAGS_INCLUDE_INTRANET_SITES','LongWord').SetUInt( $00000010);
16313: const 'ZAFLAGS_NO_UI','LongWord').SetUInt( $00000020);
16314: const 'ZAFLAGS_SUPPORTS_VERIFICATION','LongWord').SetUInt( $00000040);
16315: const 'ZAFLAGS_UNC_AS_INTRANET','LongWord').SetUInt( $00000080);
16316: const 'ZAFLAGS_USE_LOCKED_ZONES','LongWord').SetUInt( $00010000);
16317: //PZoneAttributes', 'TZoneAttributes // will not work';
16318: _ZONEATTRIBUTES', 'record cbSize:ULONG;szDisplayName:array [0..260-1] of Char;szDescription:array [0..200-1] of Char;szIconPath: array [0..260 - 1] of char; dwTemplateMinLevel: DWORD; dwTemplateRecommended: DWORD; dwTemplateCurrentLevel: DWORD; dwFlags: DWORD; end');
16319: { _ZONEATTRIBUTES = packed record
16320:   cbSize: ULONG;
16321:   szDisplayName: array [0..260 - 1] of WideChar;
16322:   szDescription: array [0..200 - 1] of WideChar;
16323:   szIconPath: array [0..260 - 1] of WideChar;
16324:   dwTemplateMinLevel: DWORD;
16325:   dwTemplateRecommended: DWORD;
16326:   dwTemplateCurrentLevel: DWORD;
16327:   dwFlags: DWORD;
16328: end; }
16329: TZoneAttributes', '_ZONEATTRIBUTES');
16330: ZONEATTRIBUTES', '_ZONEATTRIBUTES');
16331: const 'URLZONEREG_DEFAULT','LongInt').SetInt( 0);
16332: const 'URLZONEREG_HKLM','LongInt').SetInt( URLZONEREG_DEFAULT + 1);
16333: const 'URLZONEREG_HKCU','LongInt').SetInt( URLZONEREG_HKLM + 1);
16334: SIRegister_IInternetZoneManager(CL);

```

```

16335:  SIRegister_IInternetZoneManagerEx(CL);
16336:  const 'SOFTDIST_FLAG_USAGE_EMAIL','LongWord').SetUInt( $00000001);
16337:  const 'SOFTDIST_FLAG_USAGE_PRECACHE','LongWord').SetUInt( $00000002);
16338:  const 'SOFTDIST_FLAG_USAGE_AUTOINSTALL','LongWord').SetUInt( $00000004);
16339:  const 'SOFTDIST_FLAG_DELETE_SUBSCRIPTION','LongWord').SetUInt( $00000008);
16340:  const 'SOFTDIST_ASTATE_NONE','LongWord').SetUInt( $00000000);
16341:  const 'SOFTDIST_ASTATE_AVAILABLE','LongWord').SetUInt( $00000001);
16342:  const 'SOFTDIST_ASTATE_DOWNLOADED','LongWord').SetUInt( $00000002);
16343:  const 'SOFTDIST_ASTATE_INSTALLED','LongWord').SetUInt( $00000003);
16344:  //PCodeBaseHold', '^TCodeBaseHold // will not work');
16345:  _tagCODEBASEHOLD', 'record cbSize : ULONG; szDistUnit : LPWSTR;
16346:    +'szCodeBase : LPWSTR; dwVersionMS : DWORD; dwVersionLS : DWORD; dwStyle : DWORD; end');
16347:  TCodeBaseHold', '_tagCODEBASEHOLD');
16348:  CODEBASEHOLD', '_tagCODEBASEHOLD');
16349:  //PSoftDistInfo', '^TSoftDistInfo // will not work');
16350:  _tagSOFTDISTINFO', 'record cbSize : ULONG; dwFlags : DWORD; dwAdr
16351:    +'State : DWORD; szTitle : LPWSTR; szAbstract : LPWSTR; szHREF : LPWSTR; dwI
16352:    +'nstalledVersionMS : DWORD; dwInstalledVersionLS : DWORD; dwUpdateVersionMS'
16353:    +' : DWORD; dwUpdateVersionLS : DWORD; dwAdvertisedVersionMS : DWORD; dwAdv
16354:    +'rtisedVersionLS : DWORD; dwReserv : DWORD; end');
16355:  TSoftDistInfo', '_tagSOFTDISTINFO');
16356:  SOFTDISTINFO', '_tagSOFTDISTINFO');
16357:  SIRegister_ISoftDistExt(CL);
16358:  Function GetSoftwareUpdateInfo( szDistUnit : LPCWSTR; var dsi : TSoftDistInfo ) : HResult');
16359:  Function SetSoftwareUpdateAdvertisementState(szDistUnit: LPCWSTR; dwAdState, dwAdvertisedVersionMS,
dwAdvertisedVersionLS : DWORD) : HResult';
16360:  SIRegister_IDataFilter(CL);
16361:  //PProtocolFilterData', '^TProtocolFilterData // will not work');
16362:  _tagPROTOCOLFILTERDATA', 'record cbSize : DWORD; ProtocolSink :
16363:    +'IInternetProtocolSink; Protocol : IInternetProtocol; Unk : IUnknown; dwFil
16364:    +'terFlags : DWORD; end');
16365:  TProtocolFilterData', '_tagPROTOCOLFILTERDATA');
16366:  PROTOCOLFILTERDATA', '_tagPROTOCOLFILTERDATA');
16367:  //PDataInfo', '^TDataInfo // will not work');
16368:  _tagDATAINFO', 'record ulTotalSize : ULONG; ulavrPacketSize : UL
16369:    +'ONG; ulConnectSpeed : ULONG; ulProcessorSpeed : ULONG; end');
16370:  TDataInfo', '_tagDATAINFO');
16371:  DATAINFO', '_tagDATAINFO');
16372:  SIRegister_IEncodingFilterFactory(CL);
16373:  Function IsLoggingEnabled( pszUrl : PChar ) : BOOL';
16374:  //Function IsLoggingEnabledA( pszUrl : PAnsiChar ) : BOOL';
16375:  //Function IsLoggingEnabledW( pszUrl : PWideChar ) : BOOL';
16376:  //PHitLoggingInfo', 'THitLoggingInfo // will not work');
16377:  _tagHIT_LOGGING_INFO', 'record dwStructSize : DWORD; lpszLoggedU
16378:    +'rlName : LPSTR; StartTime : TSystemTime; EndTime : TSystemTime; lpszExtendedInfo : LPSTR; end');
16379:  THitLoggingInfo', '_tagHIT_LOGGING_INFO');
16380:  HIT_LOGGING_INFO', '_tagHIT_LOGGING_INFO');
16381:  Function WriteHitLogging( const Logginginfo : THitLoggingInfo ) : BOOL';
16382: end;
16383:
16384: procedure SIRegister_DFFUUtils(CL: TPPascalCompiler);
16385: begin
16386:  Procedure reformatMemo( const m : TCustomMemo );
16387:  Procedure SetMemoMargins( m : TCustomMemo; const L, T, R, B : integer );
16388:  Procedure MoveToTop( memo : TMemo );
16389:  Procedure ScrollToTop( memo : TMemo );
16390:  Function LineNumberClicked( memo : TMemo ) : integer';
16391:  Function MemoClickedLine( memo : TMemo ) : integer';
16392:  Function ClickedMemoLine( memo : TMemo ) : integer';
16393:  Function MemoLineClicked( memo : TMemo ) : integer';
16394:  Function LinePositionClicked( Memo : TMemo ) : integer');
16395:  Function ClickedMemoPosition( memo : TMemo ) : integer');
16396:  Function MemoPositionClicked( memo : TMemo ) : integer');
16397:  Procedure AdjustGridSize( grid : TDrawGrid );
16398:  Procedure DeleteGridRow( Grid : TStringGrid; const ARow : integer );
16399:  Procedure InsertgridRow( Grid : TStringGrid; const ARow : integer );
16400:  Procedure Sortgrid( Grid : TStringGrid; const SortCol : integer );
16401:  Procedure Sortgridi(Grid:TStringGrid; const SortCol : integer; sortascendant : boolean);
16402:  Procedure sortstrDown( var s : string );
16403:  Procedure sortstrUp( var s : string );
16404:  Procedure rotatestrleft( var s : string );
16405:  Function dffstrtofloatdef( s : string; default : extended ) : extended';
16406:  Function deblank( s : string ) : string';
16407:  Function IntToBinaryString( const n : integer; MinLength : integer ) : string';
16408:  Procedure FreeAndClearListBox( C : TListBox );
16409:  Procedure FreeAndClearMemo( C : TMemo );
16410:  Procedure FreeAndClearStringList( C : TStringList );
16411:  Function dffgetfilesize( f : TSearchrec ) : int64';
16412: end;
16413:
16414: procedure SIRegister_MathsLib(CL: TPPascalCompiler);
16415: begin
16416:  CL.AddTypeS('intset', 'set of byte');
16417:  TPoint64', 'record x : int64; y : int64; end');
16418:  Function GetNextPandigital( size : integer; var Digits : array of integer ) : boolean';
16419:  Function IsPolygonal( T : int64; var rank : array of integer ) : boolean';
16420:  Function GeneratePentagon( n : integer ) : integer';
16421:  Function IsPentagon( p : integer ) : boolean';
16422:  Function isSquare( const N : int64 ) : boolean';

```

```

16423: Function isCube( const N : int64 ) : boolean');
16424: Function isPalindrome( const n : int64 ) : boolean');
16425: Function isPalindromel( const n : int64; var len : integer ) : boolean');
16426: Function GetEulerPhi( n : int64 ) : int64');
16427: Function dffIntPower( a, b : int64 ) : int64');
16428: Function IntPowerl( a : extended; b : int64 ) : extended');
16429: Function gcd2( a, b : int64 ) : int64');
16430: Function GCDMany( A : array of integer ) : integer');
16431: Function LCMMany( A : array of integer ) : integer');
16432: Procedure ContinuedFraction(A: array of int64;const wholepart:integer;var numerator,denominator:int64);
16433: Function dffFactorial( n : int64 ) : int64');
16434: Function digitcount( n : int64 ) : integer');
16435: Function nextpermute( var a : array of integer ) : boolean');
16436: Function convertfloattofractionstring(N:extended; maxdenom:integer; multipleof : boolean) : string');
16437: Function convertStringToDecimal( s : string; var n : extended ) : Boolean');
16438: Function InttoBinaryStr( nn : integer ) : string');
16439: Function StrToAngle( const s : string; var angle : extended ) : boolean');
16440: Function AngleToStr( angle : extended ) : string');
16441: Function deg2rad( deg : extended ) : extended');
16442: Function rad2deg( rad : extended ) : extended');
16443: Function GetLongToMercProjection( const long : extended ) : extended');
16444: Function GetLatToMercProjection( const Lat : Extended ) : Extended');
16445: Function GetMercProjectionToLong( const ProjLong : extended ) : extended');
16446: Function GetMercProjectionToLat( const ProjLat : extended ) : extended');
16447: SIRegister_TPrimes(CL);
16448: //RIRegister_TPrimes(CL);
16449: //CL.AddConstantN('deg','LongInt').SetInt( char( 176));
16450: CL.AddConstantN('minmark','LongInt').SetInt(( 180));
16451: Function Random64( const N : Int64 ) : Int64;');
16452: Procedure Randomize64');
16453: Function Random64l : extended;');
16454: Procedure GetParens( Variables : string; OpChar : char; var list : TStringlist)//DFFUtils
16455: end;
16456:
16457: procedure SIRegister_UGeometry(CL: TPSPPascalCompiler);
16458: begin
16459:   TrealPoint', 'record x : extended; y : extended; end');
16460:   Tline', 'record pl : TPoint; p2 : TPoint; end');
16461:   TRealLine', 'record pl : TRealPoint; p2 : TRealPoint; end');
16462:   TCircle', 'record cx : integer; cy : integer; r : integer; end');
16463:   TRealCircle', 'record cx : extended; cy : extended; r : extended; end');
16464:   PPResult', '( PPOutside, PPInside, PPVertex, PPEdge, PPError )');
16465:   Function realpoint( x, y : extended ) : TRealPoint');
16466:   Function dist( const pl, p2 : TrealPoint ) : extended');
16467:   Function intdist( const pl, p2 : TPoint ) : integer');
16468:   Function dffline( const pl, p2 : TPoint ) : Tline');
16469:   Function Line1( const pl, p2 : TRealPoint ) : TRealline');
16470:   Function dffCircle( const cx, cy, R : integer ) : TCircle');
16471:   Function Circle1( const cx, cy, R : extended ) : TRealCircle');
16472:   Function GetTheta( const L : TLine ) : extended');
16473:   Function GetTheta1( const pl, p2 : TPoint ) : extended');
16474:   Function GetTheta2( const pl, p2 : TRealPoint ) : extended');
16475:   Procedure Extendline( var L : TLine; dist : integer );
16476:   Procedure Extendline1( var L : TRealLine; dist : extended );
16477:   Function Linesintersect( line1, line2 : TLine ) : boolean');
16478:   Function ExtendedLinesIntersect( Line1,Line2:TLine; const extendlines:bool;var IP:TPoint ):bool;
16479:   Function ExtendedLinesIntersect1( const Line1,Line2:TLine;const extendlines:bool;var IP:TRealPoint ):bool;
16480:   Function Intersect( L1, L2 : TLine; var pointonborder : boolean; var IP : TPoint ) : boolean');
16481:   Function PointPerpendicularLine( L : TLine; P : TPoint ) : TLine');
16482:   Function PerpDistance( L : TLine; P : TPoint ) : Integer');
16483:   Function AngledLineFromLine( L : TLine; P : TPoint; Dist : extended; alpha : extended ) : TLine');
16484:   Function AngledLineFromLine1(L:TLine;P:TPoint;Dist:extended;alpha:extended;useScreenCoordinates:bool):TLine;
16485:   Function PointInPoly( const p : TPoint; Points : array of TPoint ) : PPResult');
16486:   Function PolygonArea(const points:array of TPoint;const screencoordinates:boolean;var
16487:   Procedure InflatePolygon(const points:array of Tpoint; var points2:array of TPoint;var area:integer;
16488:   const screenCoordinates : boolean; const inflateby : integer)');
16489:   Function PolyBuiltClockwise(const points:array of TPoint;const screencoordinates:boolean):bool;
16490:   Function DegtоРad( d : extended ) : extended');
16491:   Function RadtoDeg( r : extended ) : extended');
16492:   Procedure TranslateLeftTo( var L : TLine; newend : TPoint );
16493:   Procedure TranslateLeftTo1( var L : TrealLine; newend : TrealPoint );
16494:   Procedure RotateRightEndBy( var L : TLine; alpha : extended );
16495:   Procedure RotateRightEndTo( var L : TLine; alpha : extended );
16496:   Function CircleCircleIntersect( c1, c2 : TCircle; var IP1, Ip2 : TPoint ) : boolean');
16497:   Function CircleCircleIntersect1( c1, c2 : TRealCircle; var IP1, Ip2 : TRealPoint ) : boolean');
16498:   Function PointCircleTangentLines( const C : TCircle; const P : TPoint; var L1, L2 : TLine ) : boolean');
16499:   Function CircleCircleExtTangentLines(C1,C2:TCircle;var L1,L2,PL1,PL2,TL1,TL2:TLine):Bool;
16500: end;
16501:
16502:
16503: procedure SIRegister_UAstronomy(CL: TPSPPascalCompiler);
16504: begin
16505:   TCoordType', '( ctUnknown, ctAzAlt, ctEclLonLat, ctRADecl, ctHADecl, ctGallLonLat )');
16506:   TDType', '( ttLocal, ttUT, ttGST, ttLST )');
16507:   TRPoint', 'record x : extended; y : extended; CoordType : TCoordType; end');
16508:   TSunrec', 'record TrueEclLon:extended;
16509:   AppEclLon:extended; AUDistance:extended;TrueHADecl:TRPoint;TrueRADecl : TRPoint;
16510:   TrueAzAl:TRPoint;AppHADecl:TRPoint;AppRADecl:TRPoint;AppAzAlt:TRpoint;SunMeanAnomaly:extended;end';

```

```

16509: TMoonRec', 'record ECLonLat : TRPoint; AZAlt : TRPoint; RADecl :'
16510:   +' TRPoint; HorizontalParallax : extended; AngularDiameter : extended; KMtoE'
16511:   +'arth : extended; Phase : extended; end');
16512: TMoonEclipseAdd', 'record UmbralStartTime : TDatetime; UmbralEnd'
16513:   +'Time : TDatetime; TotalStartTime : TDatetime; TotalEndTime : TDatetime; end');
16514: TEclipseRec', 'record Msg : string; Status : integer; FirstConta'
16515:   +'ct : TDatetime; LastContact : TDatetime; Magnitude:Extended; MaxEclipseUTime:TDatetime; end');
16516: TPlanet', '( MERCURY, VENUS, MARS, JUPITER, SATURN, URANUS, NEPTUNE, PLUTO )');
16517: TPlanetRec', 'record AsOf : TDatetime; Name : string; MeanLon :'
16518:   +'extended; MeanLonMotion : extended; LonOfPerihelion : extended; Eccentricity'
16519:   +'ty : extended; Inclination : extended; LonAscendingNode : extended; SemiMajorAxis'
16520:   +'jorAxis : extended; AngularDiameter : extended; Magnitude : extended; end');
16521: TPlanetLocRec', 'record PlanetBaseData : TPlanetRec; HelioCentricLonLat : TRPoint; RadiusVector :'
16522:   +'extended; UncorrectedEarthDistance : extended; GeoEclLonLat : TRPoint; CorrectedEarthDistance:extended;
16523:   +'ApparentRaDecl:TRPoint; end');
16524: SIRegister_TAstronomy(CL);
16525: Function AngleToStr( angle : extended ) : string');
16526: Function StrToAngle( s : string; var angle : extended ) : boolean';
16527: Function HoursToStr24( t : extended ) : string');
16528: Function RPoint( x, y : extended ) : TRPoint');
16529: Function getStimenename( t : TDTType ) : string');
16530: end;
16531: procedure SIRegister_UCardComponentV2(CL: TPSPascalCompiler);
16532: begin
16533:   TCardValue', 'Integer');
16534:   TCardSuit', '( Spades, Diamonds, Clubs, Hearts )');
16535:   TShortSuit', '( cardS, cardD, cardC, cardH )');
16536:   Type(TDecks.Standard1,Standard2,Fishes1,Fishes2,Beach,Leaves1,Leaves2,Robot,Roses,Shell,Castle,Hand);
16537:   SIRegister_TCard(CL);
16538:   SIRegister_TDeck(CL);
16539: end;
16540: procedure SIRegister_UTGraphSearch(CL: TPSPascalCompiler);
16541: begin
16542:   tMethodCall', 'Procedure');
16543:   tVerboseCall', 'Procedure ( s : string)');
16544:   // PTEdge', '^TEdge // will not work');
16545:   TEdge', 'record FromNodeIndex : Integer; ToNodeIndex : Integer; '
16546:   +'Weight : integer; work : integer; Len : integer; Highlight : boolean; end');
16547:   SIRegister_TNode(CL);
16548:   SIRegister_TGraphList(CL);
16549: end;
16550: procedure SIRegister_UParser10(CL: TPSPascalCompiler);
16551: begin
16552:   ParserFloat', 'extended');
16553:   //PParserFloat', '^ParserFloat // will not work');
16554:   TDFFToken', '( variab, constant, minus, sum, diff, prod, divis, mod'
16555:   +'ulo, IntDiv, IntDIVZ, integerpower, realpower, square, third, fourth, FuncOneVar, FuncTwoVar ');
16556:   //POperation', '^Operation // will not work');
16557:   TDFFOperation', 'record Arg1 : PParserFloat; Arg2 : PParserFloat; D'
16558:   +'est : PParserFloat; NextOperation : POperation; Operation : TMathProcedure'
16559:   +'; Token : TDFFToken; end');
16560:   TMathProcedure', 'procedure(AnOperation: TDFFOperation)');
16561:   (CL.FindClass('TOBJECT'), 'EMathParserError');
16562:   CL.FindClass('TOBJECT', 'ESyntaxError');
16563:   (CL.FindClass('TOBJECT'), 'EExpressionHasBlanks');
16564:   (CL.FindClass('TOBJECT'), 'EExpressionTooComplex');
16565:   (CL.FindClass('TOBJECT'), 'ETooManyNestings');
16566:   (CL.FindClass('TOBJECT'), 'EMissMatchingBracket');
16567:   (CL.FindClass('TOBJECT'), 'EBadName');
16568:   (CL.FindClass('TOBJECT'), 'EParseInternalError');
16569:   ('TEXParserExceptionEvent', 'Procedure ( Sender : TObject; E : Exception)');
16570:   SIRegister_TCustomParser(CL);
16571:   SIRegister_TExParser(CL);
16572: end;
16573: end;
16574: function isService: boolean;
16575: begin
16576:   result:= NOT(Application is TApplication);
16577:   {result:= Application is TServiceApplication;}
16578: end;
16579: function isApplication: boolean;
16580: begin
16581:   result:= Application is TApplication;
16582: end;
16583: //SM_REMOTESESSION = $1000
16584: function isTerminalSession: boolean;
16585: begin
16586:   result:= GetSystemMetrics(SM_REMOTESESSION) > 0;
16587: end;
16588: procedure SIRegister_cyIEUtils(CL: TPSPascalCompiler);
16589: begin
16590:   CL.AddTypeS('TwbPageSetup', 'record font : String; footer : String; header : '
16591:   +'String; margin_bottom : String; margin_left : String; margin_right : String'
16592:   +'g; margin_top : String; Print_Background : String; Shrink_To_Fit : String; end');
16593:   Function cyURLEncode( const S : string ) : string');

```

```

16596: Function MakeResourceURL( const ModuleName:string;const ResName:PChar;const ResType:PChar):string;
16597: Function MakeResourceURL1( const Module:HMODULE;const ResName:PChar;const ResType:PChar):string;
16598: Function cyColorToHtml( aColor : TColor ) : String';
16599: Function HtmlToColor( aHtmlColor : String ) : TColor';
16600: //Function GetStreamEncoding( aStream : TStream ) : TEncoding';
16601: // Function IsStreamEncodedWith( aStream : TStream; Encoding : TEncoding ) : Boolean';
16602: Function AddHtmlUnicodePrefix( aHtml : String ) : String';
16603: Function RemoveHtmlUnicodePrefix( aHtml : String ) : String';
16604: Procedure GetPageSetupFromRegistry( var awbPageSetup : TwbPageSetup );
16605: Procedure SetPageSetupToRegistry( awbPageSetup : TwbPageSetup );
16606: CL.AddConstantN('IEBodyBorderless','String').SetString('none');
16607: CL.AddConstantN('IEBodySingleBorder','String').SetString('');
16608: CL.AddConstantN('IEDesignModeOn','String').SetString('On');
16609: CL.AddConstantN('IEDesignModeOff','String').SetString('Off');
16610: CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FEFF );
16611: CL.AddConstantN('cHtmlUnicodePrefixChar','Char').SetString( #$FE );
16612: end;
16613:
16614:
16615: procedure SIRегистer_UcomboV2(CL: TPSCompiler);
16616: begin
16617:   CL.AddConstantN('UMaxEntries','LongInt').SetInt( 600 );
16618:   CL.AddTypeS('TCombotype', '( Combinations, Permutations, CombinationsDown, Pe' +
16619:     +'rmutationsDown, CombinationsCoLex, CombinationsCoLexDown, PermutationsRepe' +
16620:     +'at, PermutationsWithRep, PermutationsRepeatDown, CombinationsWithrep, Comb' +
16621:     +'inationsRepeat, CombinationsRepeatDown )');
16622:   CL.AddTypeS('TByteArray64', 'array[0..600 + 1] of int64;');
16623:   SIRегистer_TComboSet(CL);
16624: end;
16625:
16626: procedure SIRегистer_cyBaseComm(CL: TPSCompiler);
16627: begin
16628:   TcyCommandType', '( ctDevelopperDefined, ctUserDefined )';
16629:   TStreamContentType', '( scDevelopperDefined, scUserDefined, scString )';
16630:   TProcOnReceiveCommand', 'Procedure ( Sender: TObject; aCommand: Word; userParam : Integer )';
16631:   TProcOnReceiveString', 'Procedure ( Sender : TObject; fromBaseCo' +
16632:     +'mmHandle : THandle; aString : String; userParam : Integer )';
16633:   TProcOnReceiveMemoryStream', 'Procedure ( Sender : TObject; from' +
16634:     +'BaseCommHandle : THandle; aStream : TMemoryStream; userParam : Integer )';
16635:   SIRегистer_TcyBaseComm(CL);
16636:   CL.AddConstantN('MsgCommand','LongInt').SetInt( WM_USER + 1 );
16637:   CL.AddConstantN('MsgResultOk','LongInt').SetInt( 99 );
16638:   Function ValidatefileMappingName( aName : String ) : String';
16639:   procedure makeCaption(leftSide, Rightside:string; form:TForm);
16640: end;
16641:
16642: procedure SIRегистer_cyDERUtils(CL: TPSCompiler);
16643: begin
16644:   CL.AddTypeS('DERString', 'String');
16645:   CL.AddTypeS('DERChar', 'Char');
16646:   CL.AddTypeS('TElementsType', '( etText, etExpressionKeyWord, etNumbers, etInt' +
16647:     +'eger, etFloat, etPercentage, etwebSite, etWebMail, etMoney, etDate, etTextLine, etParagraph )');
16648:   CL.AddTypeS('TElementsTypes', 'set of TEelementsType');
16649:   CL.AddTypeS('DERNString', 'String');
16650:   const DERDecimalSeparator','String').SetString( '.' );
16651:   const DERDefaultChars','String')('+@/%-
16652:   ..:0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ');
16653:   const DERNDefaultChars','String').SetString( '/%.0123456789abcdefghijklmnopqrstuvwxyz' );
16654:   Function isValidWebMailChar( aChar : Char ) : Boolean';
16655:   Function isValidwebSite( aStr : String ) : Boolean';
16656:   Function isValidWebMail( aStr : String ) : Boolean';
16657:   Function ValidateDate( aDERStr : DERString; var RsltFormat : String ) : Boolean';
16658:   Function DERStrToDate( aDERStr, aFormat : String ) : TDate';
16659:   Function IsDERChar( aChar : Char ) : Boolean';
16660:   Function IsDERDefaultChar( aChar : Char ) : Boolean';
16661:   Function IsDERMoneyChar( aChar : Char ) : Boolean';
16662:   Function IsDERExceptionChar( aChar : Char ) : Boolean';
16663:   Function IsDERSymbols( aDERString : String ) : Boolean';
16664:   Function StringToDERCharSet( aStr : String ) : DERString';
16665:   Function IsDERNDefaultChar( aChar : Char ) : Boolean';
16666:   Function IsDERNChar( aChar : Char ) : Boolean';
16667:   Function DERToDERNCharset( aDERStr : DERString ) : DERNString';
16668:   Function DERExtractwebSite( aDERStr : DERString; SmartRecognition : Boolean ) : String';
16669:   Function DERExtractWebMail( aDERStr : DERString ) : String';
16670:   Function DERExtractPhoneNr( aDERStr : DERString ) : String';
16671:   Function DERExecute(aDERStr:DERString;SmartNumbersRecognition,
SmartWebsiteRecognition:Boolean):TElementsType;
16672:   Function DERexecutel(aDERStr : DERString; var RsltNumbers, RsltInteger, RsltFloat, RsltPercentage,
RsltwebSite, RsltWebMail,RsltMoney,RsltDate:String;SmartNumbersRecognition,
SmartWebsiteRecognition:Boolean):TElementsType;
16673:   Function RetrieveElementValue( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition: Boolean;
aElementsType : TElementsType ) : String';
16674:   Procedure RetrieveElementValue1( aStr : String; SmartNumbersRecognition, SmartWebsiteRecognition :
Boolean; var RsltDERStr : DERString; var RsltElementType : TElementsType );';
16675: end;
16676:
16677: procedure SIRегистer_cyImage(CL: TPSCompiler);
16678: begin

```

```

16679: pRGBQuadArray', '^TRGBQuadArray // will not work');
16680: Function BitmapsCompare(Bmp1:TBitmap;Bmp2:TBitmap;FirstCol,LastCol,FirstRow,LastRow:Int):Integer');
16681: Procedure BitmapSetPercentBrightness( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean)');
16682: Procedure BitmapSetPixelsBrightness( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean)');
16683: Procedure BitmapSetPercentContrast( Bmp : TBitmap; IncPercent : Integer; RefreshBmp : Boolean)');
16684: Procedure BitmapSetPixelsContrast( Bmp : TBitmap; IncPixels : Integer; RefreshBmp : Boolean)');
16685: Procedure BitmapNegative( Bmp : TBitmap; RefreshBmp : Boolean)');
16686: Procedure BitmapModifyRGB( Bmp:TBitmap;IncRed:Integer;IncGreen:Int;IncBlue:Int;RefreshBmp:Bool);
16687: Procedure BitmapReplaceColor( Bmp : TBitmap; OldColor : TColor; NewColor : TColor; RangeRed, RangeGreen,
RangeBlue : Word; SingleDestinationColor : Boolean; RefreshBmp : Boolean);');
16688: Procedure BitmapReplaceColor1( Bmp:TBitmap; OldColor:TColor;NewColor:TColor;PercentRange1Red,
PercentRange1Green, PercentRange1Blue:Extended;PercentRange2Red,PercentRange2Green,
PercentRange2Blue:Double;SingleDestinationColor: Boolean;RefreshBmp:Bool;');
16689: Procedure BitmapReplaceColors( Bmp: TBitmap; Array_OldPalette, Array_NewPalette : array of TColor;
SingleDestinationColor, RefreshBmp : Boolean)');
16690: Procedure BitmapResize(SourceBmp TBitmap;DestinationBmp:TBitmap;Percent:Extended;RefreshBmp : Boolean)');
16691: Procedure BitmapRotate( Bmp : TBitmap; AngleDegree : Word; AdjustSize : Boolean; BkColor : TColor)');
16692: Procedure BitmapBlur(SourceBmp:TBitmap;DestinationBmp:TBitmap;Pixels:Word;Percent:Integer;RefreshBmp:Bool;
16693: Procedure GraphicMirror(Source:TGraphic;Destination:TCanvas;Left,Top:Integer;Horizontal,Vertical:Bool;
16694: Procedure GraphicMirror1(Source:TGraphic;Destination:TBitmap;Horizontal:Boolean;Vertical:Boolean)');
16695: Function BitmapCreate(BmpWidth:Integer;BmpHeight:Integer;BgColor:TColor;PixelFormat:TPixelFormat):TBitmap;
16696: Procedure BitmapSaveToJpegFile( Bmp : TBitmap; FileName : String; QualityPercent : Word)');
16697: Procedure JpegSaveToBitmapFile( JPEG : TJPEGImage; FileName : String)';
16698: end;
16699:
16700: procedure SIRegister_WaveUtils(CL: TPSPascalCompiler);
16701: begin
16702:   TMS2StrFormat', '( msHMSh, msHMS, msMS, msSh, msS, msAh,msA )');
16703:   TPCMChannel', '( cMono, cStereo )');
16704:   TPCMSamplesPerSec', '( ss8000Hz, ss11025Hz, ss22050Hz, ss44100Hz, ss48000Hz )');
16705:   TPCMBitsPerSample', '( bs8Bit, bs16Bit )');
16706:   TPCMFormat', '( nonePCM, Mono8Bit8000Hz, Stereo8bit8000Hz, Mono16b'
16707:     +'6bit8000Hz, Stereo16bit8000Hz, Mono8bit11025Hz, Stereo8bit11025Hz, Mono16b'
16708:     +'it11025Hz, Stereo16bit11025Hz, Mono8bit22050Hz, Stereo8bit22050Hz, Mono16b'
16709:     +'it22050Hz, Stereo16bit22050Hz, Mono8bit44100Hz, Stereo8bit44100Hz, Mono16b'
16710:     +'it44100Hz, Stereo16bit44100Hz, Mono8bit48000Hz, Stereo8bit48000Hz, Mono16b'
16711:     +'it48000Hz, Stereo16bit48000Hz )');
16712:   PWaveFormatEx', 'record wFormatTag : word; nChannels: word; nSamplesPerSec: DWORD; '
16713:     +'nAvgBytesPerSec: DWORD; nBlockAlign: Word; wBitsPerSample: Word;  ecbSiz: Word; end');
16714:   tWaveFormatEx', 'PWaveFormatEx');
16715:   HMMIO', 'Integer');
16716:   TWaveDeviceFormats', 'set of TPCMFormat');
16717:   TWaveOutDeviceSupport', '( dsVolume, dsStereoVolume, dsPitch, ds'
16718:     +'PlaybackRate, dsPosition, dsAsynchronize, dsDirectSound )');
16719:   CL.AddTypeS('TWaveOutDeviceSupports', 'set of TWaveOutDeviceSupport');
16720:   TWaveOutOption', '( woSetVolume, woSetPitch, woSetPlaybackRate )');
16721:   TWaveOutOptions', 'set of TWaveOutOption');
16722:   TStreamOwnership2', '( soReference, soOwned )');
16723:   TWaveStreamState', '( wssReady, wssReading, wssWriting, wssWritingEx )');
16724: // PRawWave', '^PRawWave // will not work');
16725:   TRawWave', 'record pData : TObject; dwSize : DWORD; end');
16726:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EWaveAudioError');
16727:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EWaveAudioSysError');
16728:   CL.AddClassN(CL.FindClass('TOBJECT'), 'EWaveAudioInvalidOperation');
16729:   TWaveAudioEvent', 'Procedure ( Sender : TObject)');
16730:   TWaveAudioGetFormatEvent', 'Procedure ( Sender : TObject; var PW'
16731:     +'aveFormat : PWaveFormatEx; var FreeIt : Boolean)');
16732:   TWaveAudioGetDataEvent', 'Function ( Sender : TObject; const Buf'
16733:     +'fer : TObject; BufferSize : DWORD; var NumLoops : DWORD) : DWORD');
16734:   TWaveAudioGetDataPtrEvent', 'Function ( Sender : TObject; var Bu'
16735:     +'ffer : TObject; var NumLoops : DWORD; var FreeIt : Boolean) : DWORD');
16736:   TWaveAudioDataReadyEvent', 'Procedure ( Sender : TObject; const '
16737:     +'Buffer : TObject; BufferSize : DWORD; var FreeIt : Boolean)');
16738:   TWaveAudioLevelEvent', 'Procedure ( Sender : TObject; Level : Integer)');
16739:   TWaveAudioFilterEvent', 'Procedure (Sender : TObject; const Buffer:TObject; BufferSize:DWORD);
16740:   GetWaveAudioInfo(mmIO:HMMIO;var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16741:   CreateWaveAudio(mmIO: HMMIO; const pWaveFormat:PWaveFormatEx;var ckRIFF,ckData:TMMCKInfo):Bool;
16742:   CloseWaveAudio( mmIO : HMMIO; var ckRIFF, ckData : TMMCKInfo)');
16743:   GetStreamWaveAudioInfo(Stream:TStream; var pWaveFormat:PWaveFormatEx;var DataSize,DataOffset:DWORD):Bool;
16744:   CreateStreamWaveAudio(Stream: TStream;const pWaveFormat:PWaveFormatEx; var ckRIFF,ckData:TMMCKInfo):HMMIO;
16745:   OpenStreamWaveAudio( Stream : TStream ) : HMMIO';
16746:   CalcWaveBufferSize( const pWaveFormat : PWaveFormatEx; Duration : DWORD) : DWORD');
16747:   GetAudioFormat( FormatTag : Word) : String');
16748:   GetWaveAudioFormat( const pWaveFormat : PWaveFormatEx) : String');
16749:   GetWaveAudioLength( const pWaveFormat : PWaveFormatEx; DataSize : DWORD) : DWORD');
16750:   GetWaveAudioBitRate( const pWaveFormat : PWaveFormatEx) : DWORD');
16751:   GetWaveAudioPeakLevel(const Data : TObject;DataSize:DWORD; const pWaveFormat:PWaveFormatEx): Integer');
16752:   InvertWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx): Boolean');
16753:   SilenceWaveAudio( const Data : TObject; DataSize : DWORD;const pWaveFormat : PWaveFormatEx): Boolean');
16754:   ChangeWaveAudioVolume(const Data:TObject;DataSize:DWORD;const pWaveFormat:PWaveFormatEx;Percent:Int):Bool;
16755:   MixWaveAudio( const RawWaves : TRawWave; Count : Integer; const pWaveFormat : PWaveFormatEx; Buffer :
TObject; BufferSize : DWORD) : Boolean');
16756:   ConvertWaveFormat( const srcFormat : PWaveFormatEx; srcData : TObject; srcDataSize : DWORD; const
dstFormat : PWaveFormatEx; var dstData : TObject; var dstDataSize : DWORD) : Boolean');
16757:   SetPCMFormat(const pWaveFormat : PWaveFormatEx; Channels : TPCMChannel; SamplesPerSec :
TPCMSamplesPerSec; BitsPerSample : TPCMBitsPerSample)');
16758:   Procedure SetPCMFormat(const pWaveFormat : PWaveFormatEx; PCMFormat : TPCMFormat)');
16759:   GetPCMFormat( const pWaveFormat : PWaveFormatEx) : TPCMFormat');
16760:   GetWaveDataPositionOffset( const pWaveFormat : PWaveFormatEx; Position : DWORD) : DWORD');

```

```

16761: MS2Str( Milliseconds : DWORD; Fmt : TMS2StrFormat ) : String';
16762: WaitForSyncObject( SyncObject : THandle; Timeout : DWORD ) : DWORD';
16763: //mmioStreamProc( lpmmIOInfo : PMMIOInfo; uMsg, lParam1, lParam2 : DWORD) : LRESULT';
16764: end;
16765:
16766: procedure SIRегистre_NamedPipes(CL: TPSPascalCompiler);
16767: begin
16768:   'DEFAULT_PIPE_BUFFER_SIZE','LongInt').SetInt( 4096 );
16769:   CL.AddConstantN('DEFAULT_PIPE_TIMEOUT','LongInt').SetInt( 5000 );
16770:   'PIPE_NAMING_SCHEME','String').SetString( '\%s(pipe)\%s' );
16771:   'WAIT_ERROR','LongWord').SetUInt( DWORD ( $FFFFFF ) );
16772:   'WAIT_OBJECT_1','LongInt').SetInt( WAIT_OBJECT_0 + 1 );
16773:   'STATUS_SUCCESS','LongWord').SetUInt( $00000000 );
16774:   'STATUS_BUFFER_OVERFLOW','LongWord').SetUInt( $80000005 );
16775:   CL.AddTypeS('TPipeDirection', '( pdir_Duplex, pdir_ClientToServer, pdir_ServerToClient )');
16776:   CL.AddTypeS('TPipeType', '( ptyp_BytByte, ptyp_MsgByte, ptyp_MsgMsg )');
16777:   CL.AddTypeS('TOverlappedResult', '( ov_Failed, ov_Pending, ov_MoreData, ov_Complete )');
16778:   SIRегистre_TNamedPipe(CL);
16779:   SIRегистre_TServerPipe(CL);
16780:   SIRегистre_TCClientPipe(CL);
16781:   CL.AddDelphiFunction('Function CalculateTimeout( aBasis : DWORD ) : DWORD');
16782:   Function HasOverlappedIoCompleted( const ov : OVERLAPPED ) : Boolean';
16783:   Function GetOverlappedPipeResult(aHandle:THandle;const ov:OVERLAPPED;var dwBytes:DWORD;bWait:Boolean): TOverlappedResult;
16784:   Function GetStreamAsText( stm : TStream ) : string';
16785:   Procedure SetStreamAsText( const aTxt : string; stm : TStream )';
16786: end;
16787:
16788: procedure SIRегистre_DPUtils(CL: TPSPascalCompiler);
16789: begin
16790:   // CL.AddTypeS('pRGBTripleArray', '^TRGBTripleArray // will not work');
16791:   SIRегистre_TThumbData(CL);
16792:   'PIC_BMP','LongInt').SetInt( 0 );
16793:   'PIC_JPG','LongInt').SetInt( 1 );
16794:   'THUMB_WIDTH','LongInt').SetInt( 60 );
16795:   'THUMB_HEIGHT','LongInt').SetInt( 60 );
16796:   Function IsEqualFile(Filename:string; Size:Integer; LastWriteTime : TDateTime): Boolean';
16797:   Procedure GetFileInfo(Filename : string; var Size : Integer; var LastWriteTime : TDateTime)';
16798:   Function ReadBitmap( Filehandle : Integer; Width, Height : Integer ) : TBitmap';
16799:   Procedure WriteBitmap( Filehandle : Integer; bmp : TBitmap )';
16800:   Function OpenPicture( fn : string; var tp : Integer ) : Integer';
16801:   Function ConvertPicture( pi : Integer; tp : Integer ) : TBitmap';
16802:   Function LoadPicture( fn : string; var w, h : Integer ) : TBitmap';
16803:   Function TurnBitmap( bmp : TBitmap; ang : Integer ) : TBitmap';
16804:   Function RotateBitmap( Bitmap : TBitmap; Direction : Integer ) : TBitmap';
16805:   Function StretchBitmap( Canvas : TCanvas; re : TRect; bmp : TBitmap ) : TRect';
16806:   Function ThumbBitmap( Bitmap : TBitmap; Width, Height : Integer ) : TBitmap';
16807:   Procedure ClearFrame( Canvas : TCanvas; Rect : TRect; Width, Height : Integer )';
16808:   Procedure FindFiles( path, mask : string; items : TStringList )';
16809:   Function LetfileName( s : string ) : string';
16810:   Function LetParentPath( path : string ) : string';
16811:   Function AddBackSlash( path : string ) : string';
16812:   Function CutBackSlash( path : string ) : string';
16813: end;
16814:
16815: procedure SIRегистre_CommonTools(CL: TPSPascalCompiler);
16816: begin
16817:   //BYTES','LongInt').SetInt( 1 );
16818:   'KBYTES','LongInt').SetInt( 1024 );
16819:   'DBG_ALIVE','LongWord').SetUInt( Integer ( $11BABEL1 ) );
16820:   'DBG_DESTROYING','LongWord').SetUInt( Integer ( $44FADE44 ) );
16821:   'DBG_GONE','LongWord').SetUInt( $99AC1D99 );
16822:   'SHELL_NS_MYCOMPUTER','String').SetString( ':::{20D04FE0-3AEA-1069-A2D8-08002B30309D}' );
16823:   SIRегистre_MakeComServerMethodsPublic(CL);
16824:   CL.AddTypeS('TSomeFileInfo', '( fi_DisplayType, fi_Application )');
16825:   Function IsFlagSet( dwTestForFlag, dwFlagSet : DWORD ) : Boolean';
16826:   Procedure TBSetFlag( const dwThisFlag : DWORD; var dwFlagSet : DWORD; aSet : Boolean )';
16827:   Function TBGetTempFolder : string';
16828:   Function TBGetTempFile : string';
16829:   Function TBGetModuleFilename : string';
16830:   Function FormatModuleVersionInfo( const aFilename : string ) : string';
16831:   Function GetVersionInfoString( const aFile, aEntry : string; aLang : WORD ) : string';
16832:   Function TBGetFileSize( aFile : string; aMultipleOf : Integer ) : Integer';
16833:   Function FormatAttribString( aAttr : Integer ) : string';
16834:   Function GetSomeFileInfo( aFile : string; aWhatInfo : TSomeFileInfo ) : string';
16835:   Function ShellRecycle( aWnd : HWND; aFileOrFolder : string ) : Boolean';
16836:   Function IsDebuggerPresent : BOOL';
16837:   Function TBNotImplemented : HRESULT';
16838: end;
16839:
16840: procedure SIRегистre_D2_VistaHelperU(CL: TPSPascalCompiler);
16841: begin
16842:   //CL.AddTypeS('OSVERSIONINFOEXA', '_OSVERSIONINFOEXA');
16843:   //CL.AddTypeS('TOSVersionInfoExA', '_OSVERSIONINFOEXA');
16844:   CL.AddTypeS('TOSVersionInfoEx', 'TOSVersionInfo');
16845:   CL.AddTypeS('TDrivesProperty', 'array[1..26] of boolean');
16846:   //TDrivesProperty = array['A'..'Z'] of boolean;
16847:   Function TBSetSystemTime( DateTime : TDateTime; DOW : word ) : boolean';
16848:   Function IsElevated : Boolean';

```

```

16849: Procedure CoCreateInstanceAsAdmin(aHWnd:HWND;const aClassID:GUID;const aIID:GUID;out aObj:TObject);
16850: CL.AddTypeS('TPasswordUsage', '({ pu_None, pu_Default, pu_Defined })');
16851: Function TrimNetResource(UNC : string) : string';
16852: Procedure GetFreeDrives( var FreeDrives : TDrivesProperty)');
16853: Procedure GetMappedDrives( var MappedDrives : TDrivesProperty)');
16854: Function MapDrive( UNCPPath : string; Drive : char; PasswordUsage : TPasswordUsage; Password : string;
UserUsage : TPasswordUsage; User : string; Comment : string) : boolean');
16855: Function UnmapDrive( Drive : char; Force : boolean) : boolean');
16856: Function TBIsWindowsVista : Boolean');
16857: Procedure SetVistaFonts( const AForm : TForm)');
16858: Procedure SetVistaContentFonts( const AFont : TFont)');
16859: Function GetProductType( var sType : String) : Boolean');
16860: Function lstrcmp( lpString1, lpString2 : PChar) : Integer');
16861: Function lstrcmpi( lpString1, lpString2 : PChar) : Integer');
16862: Function lstrcpy( lpString1, lpString2 : PChar) : PChar');
16863: Function lstrcpy( lpString1, lpString2 : PChar) : PChar');
16864: Function lstrcat( lpString1, lpString2 : PChar) : PChar');
16865: Function lstrlen( lpString : PChar) : Integer');
16866: Function GetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
TokenInformation : __Pointer; TokenInformationLength : DWORD; var ReturnLength : DWORD) : BOOL');
16867: Function SetTokenInformation( TokenHandle : THandle; TokenInformationClass : TTokenInformationClass;
TokenInformation : __Pointer; TokenInformationLength : DWORD) : BOOL');
16868: end;
16869:
16870: procedure SIRegister_dwsXPlatform(CL: TPSPascalCompiler);
16871: begin
16872:   'cLineTerminator','Char').SetString( #10);
16873:   'cLineTerminators','String').SetString( #13#10);
16874:   'INVALID_HANDLE_VALUE','LongInt').SetInt( DWORD( - 1 ) );
16875:   SIRegister_TFixedCriticalSection(CL);
16876:   SIRegister_TMultiReadSingleWrite(CL);
16877:   CL.AddDelphiFunction('Procedure SetDecimalSeparator( c : Char)');
16878:   Function GetDecimalSeparator : Char');
16879:   TCollectFileProgressEvent', 'Procedure ( const directory : String; var skipScan : Boolean)');
16880:   Procedure Collectfiles(const directory,fileMask:UnicodeString; list:TStrings;
recurseSubdirectories:Boolean; onProgress : TCollectFileProgressEvent)');
16881:   CL.AddTypeS('NativeInt', 'Integer');
16882:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16883:   CL.AddTypeS('NativeUInt', 'Cardinal');
16884:   //CL.AddTypeS('PNativeUInt', '^NativeUInt // will not work');
16885:   //CL.AddTypeS('TBytes', 'array of Byte');
16886:   CL.AddTypeS('RawByteString', 'UnicodeString');
16887:   //CL.AddTypeS('PNativeInt', '^NativeInt // will not work');
16888:   //CL.AddTypeS('PUInt64', '^UInt64 // will not work');
16889:   SIRegister_TPath(CL);
16890:   SIRegister_Tfile(CL);
16891:   SIRegister_TdwsThread(CL);
16892:   Function GetSystemMilliseconds : Int64');
16893:   Function UTCDateTime : TDateTime');
16894:   Function UnicodeFormat(const fmt:UnicodeString; const args : array of const): UnicodeString');
16895:   Function UnicodeCompareStr( const S1, S2 : UnicodeString) : Integer');
16896:   Function dwsAnsiCompareText( const S1, S2 : UnicodeString) : Integer');
16897:   Function dwsAnsiCompareStr( const S1, S2 : UnicodeString) : Integer');
16898:   Function UnicodeComparePChars( p1 : PChar; n1 : Integer; p2 : PChar; n2 : Integer) : Integer');
16899:   Function UnicodeComparePChar1( p1, p2 : PChar; n : Integer) : Integer');
16900:   Function UnicodeLowerCase( const s : UnicodeString) : UnicodeString');
16901:   Function UnicodeUpperCase( const s : UnicodeString) : UnicodeString');
16902:   Function ASCIICompareText( const s1, s2 : UnicodeString) : Integer');
16903:   Function ASCIISameText( const s1, s2 : UnicodeString) : Boolean');
16904:   Function InterlockedIncrement( var val : Integer) : Integer');
16905:   Function InterlockedDecrement( var val : Integer) : Integer');
16906:   Procedure FastInterlockedIncrement( var val : Integer)');
16907:   Procedure FastInterlockedDecrement( var val : Integer)');
16908:   Function InterlockedExchangePointer( var target : __Pointer; val : __Pointer) : __Pointer');
16909:   Procedure SetThreadName( const threadName : Char; threadID : Cardinal)');
16910:   Procedure dwsOutputDebugString( const msg : UnicodeString)');
16911:   Procedure WriteToOSEventLog(const logName,logCaption,logDetails:UnicodeString;const logRawData:String);
16912:   Function TryTextToFloat(const s:PChar;var value:Extended;const formatSettings:TFormatSettings):Bool;
16913:   Procedure VarCopy( out dest : Variant; const src : Variant)');
16914:   Function VarToUnicodeStr( const v : Variant) : UnicodeString');
16915:   Function LoadTextFromBuffer( const buf : TBytes) : UnicodeString');
16916:   Function LoadTextFromStream( aStream : TStream) : UnicodeString');
16917:   Function LoadTextFromFile( const fileName : UnicodeString) : UnicodeString');
16918:   Procedure SaveTextToUTF8File( const fileName, text : UnicodeString)');
16919:   Function OpenFileForSequentialReadOnly( const fileName : UnicodeString) : THandle');
16920:   Function OpenFileForSequentialWriteOnly( const fileName : UnicodeString) : THandle');
16921:   Procedure CloseFileHandle( hFile : THandle)');
16922:   Function FileCopy(const existing, new : UnicodeString; failIfExists : Boolean):Boolean');
16923:   Function FileMove( const existing, new : UnicodeString) : Boolean');
16924:   Function dwsFileDelete( const fileName : String) : Boolean');
16925:   Function FileRename( const oldName, newName : String) : Boolean');
16926:   Function dwsFileSize( const name : String) : Int64');
16927:   Function dwsFileDateTime( const name : String) : TDateTime');
16928:   Function DirectSet8087CW( newValue : Word) : Word');
16929:   Function DirectSetMXCSR( newValue : Word) : Word');
16930:   Function TtoObject( const T: byte) : TObject');
16931:   Function TtoPointer( const T: byte) : __Pointer');
16932:   Procedure GetMemForT(var T: byte; Size : integer)');
16933:   Function FindDelimiter( const Delimiters, S : string; StartIdx : Integer) : Integer');

```

```

16934: end;
16935:
16936: procedure SIRegister_AdSocket(CL: TPSCompiler);
16937: begin
16938:   'IPStrSize','LongInt').SetInt( 15 );
16939:   'CM_APDSOCKETMESSAGE','LongWord').SetUInt( WM_USER + $0711 );
16940:   'CM_APDSOCKETQUIT','LongWord').SetUInt( WM_USER + $0712 );
16941:   'ADWSBASE','LongInt').SetInt( 9000 );
16942:   CL.AddTypeS('TCMAPDSocketMessage', 'record Msg : Cardinal; Socket : TSocket; '
16943:     +'SelectEvent : Word; SelectError : Word; Result : Longint; end');
16944:   SIRegister_EApdSocketException(CL);
16945:   TWsMode', '( wsClient, wsServer )');
16946:   TWSNotifyEvent', 'Procedure ( Sender : TObject; Socket : TSocket )';
16947:   TWsSocketErrorEvent', 'Procedure ( Sender : TObject; Socket : TSocket; ErrorCode : Integer )');
16948:   SIRegister_TApdSocket(CL);
16949: end;
16950:
16951: procedure SIRegister_AdPort(CL: TPSCompiler);
16952: begin
16953:   SIRegister_TApdCustomComPort(CL);
16954:   SIRegister_TApdComPort(CL);
16955:   Function ComName( const ComNumber : Word ) : ShortString';
16956:   Function SearchComPort( const C : TComponent ) : TApdCustomComPort );
16957: end;
16958:
16959: procedure SIRegister_PathFunc(CL: TPSCompiler);
16960: begin
16961:   Function inAddBackslash( const S : String ) : String';
16962:   Function PathChangeExt( const Filename, Extension : String ) : String';
16963:   Function PathCharCompare( const S1, S2 : PChar ) : Boolean';
16964:   Function PathCharIsSlash( const C : Char ) : Boolean';
16965:   Function PathCharIsTrailByte( const S : String; const Index : Integer ) : Boolean';
16966:   Function PathCharLength( const S : String; const Index : Integer ) : Integer';
16967:   Function inPathCombine( const Dir, Filename : String ) : String';
16968:   Function PathCompare( const S1, S2 : String ) : Integer';
16969:   Function PathDrivePartLength( const Filename : String ) : Integer';
16970:   Function PathDrivePartLengthFix( const Filename:String;const IncludeSignificantSlash:Bool):Int;
16971:   Function inPathExpand( const Filename : String ) : String';
16972:   Function PathExtensionPos( const Filename : String ) : Integer';
16973:   Function PathExtractDir( const Filename : String ) : String';
16974:   Function PathExtractDrive( const Filename : String ) : String';
16975:   Function PathExtractExt( const Filename : String ) : String';
16976:   Function PathExtractName( const Filename : String ) : String';
16977:   Function PathExtractPath( const Filename : String ) : String';
16978:   Function PathIsRooted( const Filename : String ) : Boolean';
16979:   Function PathLastChar( const S : String ) : PChar';
16980:   Function PathLastDelimiter( const Delimiters, S : string ) : Integer';
16981:   Function PathLowercase( const S : String ) : String';
16982:   Function PathNormalizeSlashes( const S : String ) : String';
16983:   Function PathPathPartLength(const Filename:String;const IncludeSlashesAfterPath:Bool):Int;
16984:   Function PathPos( Ch : Char; const S : String ) : Integer';
16985:   Function PathStartsWith( const S, AStartsWith : String ) : Boolean';
16986:   Function PathStrNextChar( const S : PChar ) : PChar';
16987:   Function PathStrPrevChar( const Start, Current : PChar ) : PChar';
16988:   Function PathStrScan( const S : PChar; const C : Char ) : PChar';
16989:   Function inRemoveBackslash( const S : String ) : String';
16990:   Function RemoveBackslashUnlessRoot( const S : String ) : String';
16991:   Procedure PathFuncRunTests( const AlsoTestJapaneseDBCS : Boolean );
16992: end;
16993:
16994:
16995: procedure SIRegister_CmnFunc2(CL: TPSCompiler);
16996: begin
16997:   NEWREGSTR_PATH_SETUP','String').SetString( 'Software\Microsoft\Windows\CurrentVersion' );
16998:   NEWREGSTR_PATH_EXPLORER','String').SetString( NEWREGSTR_PATH_SETUP + '\Explorer' );
16999:   NEWREGSTR_PATH_SPECIAL_FOLDERS','String').SetString(NEWREGSTR_PATH_EXPLORER + '\Shell Folders' );
17000:   NEWREGSTR_PATH_UNINSTALL','String').SetString( NEWREGSTR_PATH_SETUP + '\Uninstall' );
17001:   NEWREGSTR_VAL_UNINSTALLER_DISPLAYNAME','String').SetString( 'DisplayName' );
17002:   NEWREGSTR_VAL_UNINSTALLER_COMMANDLINE','String').SetString( 'UninstallString' );
17003:   KEY_WOW64_64KEY,'LongWord').SetUInt( $0100 );
17004:   //CL.AddTypeS('TLeadByteSet', '^TLeadByteSet // will not work');
17005:   CL.AddTypeS('TLeadByteSet', 'set of Char');
17006:   SIRegister_TOneShotTimer(CL);
17007:   CL.AddTypeS('TRegView', '( rvDefault, rv32Bit, rv64Bit )');
17008:   'RegViews64Bit','LongInt').Value.ts32 := ord(rv64Bit);
17009:   Function NewFileExists( const Name : String ) : Boolean';
17010:   Function inDirExists( const Name : String ) : Boolean';
17011:   Function FileOrDirExists( const Name : String ) : Boolean';
17012:   Function IsDirectoryAndNotReparsePoint( const Name : String ) : Boolean';
17013:   Function GetIniString( const Section, Key : String; Default:String; const Filename:String): String';
17014:   Function GetIniInt(const Section,Key:String;const Default,Min,Max:Longint;const Filename:String):Longint;
17015:   Function GetIniBool( const Section,Key:String; const Default:Boolean;const Filename:String ) : Boolean';
17016:   Function IniKeyExists( const Section, Key, Filename : String ) : Boolean';
17017:   Function IsIniSectionEmpty( const Section, Filename : String ) : Boolean';
17018:   Function SetIniString( const Section, Key, Value, Filename : String ) : Boolean';
17019:   Function SetIniInt(const Section,Key:String;const Value: Longint;const Filename: String):Boolean';
17020:   Function SetIniBool(const Section,Key:String;const Value: Boolean;const Filename: String):Boolean';
17021:   Procedure DeleteIniEntry( const Section, Key, Filename : String );
17022:   Procedure DeleteIniSection( const Section, Filename : String );

```

```

17023: Function GetEnv( const EnvVar : String ) : String';
17024: Function GetCmdTail : String');
17025: Function GetCmdTailEx( StartIndex : Integer ) : String');
17026: Function NewParamCount : Integer');
17027: Function NewParamStr( Index : Integer ) : string');
17028: Function AddQuotes( const S : String ) : String');
17029: Function RemoveQuotes( const S : String ) : String');
17030: Function inGetShortName( const LongName : String ) : String');
17031: Function inGetWinDir : String');
17032: Function inGetSystemDir : String');
17033: Function GetSysWow64Dir : String');
17034: Function GetSysNativeDir( const IsWin64 : Boolean ) : String');
17035: Function inGetTempDir : String');
17036: Function StringChange( var S : String; const FromStr,ToStr : String ) : Integer');
17037: Function StringChangeEx(var S:String;const FromStr,ToStr:String;const SupportDBCS:Bool):Int;
17038: Function AdjustLength( var S : String; const Res : Cardinal ) : Boolean');
17039: Function UsingWinNT : Boolean');
17040: Function ConvertConstPercentStr( var S : String ) : Boolean');
17041: Function ConvertPercentStr( var S : String ) : Boolean');
17042: Function ConstPos( const Ch : Char; const S : String ) : Integer');
17043: Function SkipPastConst( const S : String; const Start : Integer ) : Integer');
17044: Function RegQueryStringValue( H : HKEY; Name : PChar; var ResultStr : String ) : Boolean');
17045: Function RegQueryMultiStringValue( H : HKEY; Name : PChar; var ResultStr : String ) : Boolean');
17046: Function RegValueExists( H : HKEY; Name : PChar ) : Boolean');
17047: Function RegCreateKeyExView( const RegView:TRegView;hKey:HKEY;lpSubKey:PChar;Reserved:DWORD;lpClass:PChar;
dwOptions : DWORD; samDesired : REGSAM; lpSecurityAttributes : TObject;
+ 'var phkResult : HKEY; lpdwDisposition : DWORD) : Longint');
17049: Function RegOpenKeyExView(const
RegView:TRegView;hKey:HKEY;lpSubKey:PChar;ulOptions:DWORD;samDesired:REGSAM;var phkResult : HKEY ) :
Longint );
17050: Function RegDeleteKeyView( const RegView : TRegView; const Key : HKEY; const Name : PChar ) : Longint );
17051: Function RegDeleteKeyIncludingSubkeys(const RegView:TRegView;const Key:HKEY;const Name:PChar):Longint;
17052: Function RegDeleteKeyIfEmpty(const RegView:TRegView;const RootKey:HKEY;const SubKeyName:PChar):Longint );
17053: Function GetShellFolderPath( const FolderID : Integer ) : String');
17054: Function IsAdminLoggedOn : Boolean');
17055: Function IsPowerUserLoggedOn : Boolean');
17056: Function IsMultiByteString( const S : AnsiString ) : Boolean');
17057: Function FontExists( const FaceName : String ) : Boolean');
17058: //Procedure FreeAndNil( var Obj );
17059: Function SafeLoadlibrary( const Filename : String; ErrorMode : UINT ) : HMODULE');
17060: Function GetUILanguage : LANGID');
17061: Function RemoveAccelChar( const S : String ) : String');
17062: Function GetTextWidth(const DC : HDC; S : String; const Prefix:Boolean):Integer');
17063: Function AddPeriod( const S : String ) : String');
17064: Function GetExceptMessage : String');
17065: Function GetPreferredUIFont : String');
17066: Function IsWildcard( const Pattern : String ) : Boolean');
17067: Function WildcardMatch( const Text, Pattern : PChar ) : Boolean');
17068: Function IntMax( const A, B : Integer ) : Integer');
17069: Function Win32ErrorString( ErrorCode : Integer ) : String');
17070: Procedure GetLeadBytes( var ALeadBytes : TLeadByteSet );
17071: Function inCompareMem( P1, P2 : TObject; Length : Integer ) : Boolean');
17072: Function DeletedDirTree( const Dir : String ) : Boolean');
17073: Function SetNTFSCompression(const FileOrDir: String; Compress : Boolean) : Boolean');
17074: Procedure AddToWindowMessageFilterEx( const Wnd : HWND; const Msg : UINT );
17075: // CL.AddTypes('TSysCharSet', 'set of AnsiChar');
17076: Function inCharInSet( C : Char; const CharSet : TSysCharSet ) : Boolean');
17077: Function ShutdownBlockReasonCreate( Wnd : HWND; const Reason : String ) : Boolean');
17078: Function ShutdownBlockReasonDestroy( Wnd : HWND ) : Boolean');
17079: Function TryStrToBoolean( const S : String; var BoolResult : Boolean ) : Boolean');
17080: Procedure WaitMessageWithTimeout( const Milliseconds : DWORD );
17081: Function MoveFileReplace(const ExistingFileName, NewFileName : String ) : Boolean');
17082: Procedure TryEnableAutoCompleteFileSystem( Wnd : HWND );
17083: end;
17084:
17085: procedure SIRegister_CmnFunc(CL: TPPascalCompiler);
17086: begin
17087:   SIRegister_TWindowDisabler(CL);
17088:   TMsgBoxType', '( mbInformation, mbConfirmation, mbError, mbCriticalError )');
17089:   TMsgBoxCallbackFunc', 'procedure(const Flags:LongInt;const After:Bool;const Param:LongInt);
17090:   Procedure UpdateHorizontalExtent( const ListBox : TCustomListBox );
17091:   Function MinimizePathName(const Filename:String; const Font:TFont;MaxLen:Integer):String;
17092:   Function AppMessageBox( const Text, Caption : PChar; Flags : Longint ) : Integer');
17093:   Function MsgBoxP( const Text,Caption: PChar; const Typ: TMMsgBoxType;const Buttons:Cardinal):Int;
17094:   Function inMsgBox(const Text,Caption:String; const Typ: TMMsgBoxType;const Buttons:Cardinal):Int;
17095:   Function MsgBoxFmt( const Text : String; const Args : array of const; const Caption : String; const Typ : TMMsgBoxType; const Buttons : Cardinal ) : Integer');
17096:   Procedure ReactivateTopWindow');
17097:   Procedure SetMessageBoxCaption( const Typ : TMMsgBoxType; const NewCaption : PChar );
17098:   Procedure SetMessageBoxRightToLeft( const ARightToLeft : Boolean );
17099:   Procedure SetMessageBoxCallbackFunc(const AFunc : TMMsgBoxCallbackFunc; const AParam : LongInt );
17100: end;
17101:
17102:
17103: {A simple Oscilloscope using TWaveIn class.
17104: More info at http://www.delphiforfun.org/programs/oscilloscope.htm }
17105: uses
17106:   Forms,
17107:   U_Oscilloscope4 in 'U_Oscilloscope4.pas' {frmMain},

```

```

17108: ufrmOscilloscope4 in 'ufrmOscilloscope4.pas' {frmOscilloscope: TFrame},
17109: uColorFunctions in 'uColorFunctions.pas',
17110: AMixer in 'AMixer.pas',
17111: uSettings in 'uSettings.pas',
17112: UWavein4 in 'UWavein4.pas',
17113: U_Spectrum4 in 'U_Spectrum4.pas' {Form2},
17114: ufrmInputControl4 in 'ufrmInputControl4.pas' {frmInputControl: TFrame};
17115:
17116:
17117: Functions_max hex in the box maxbox
17118: functionslist.txt
17119: FunctionsList1 3.9.9.86/88/91/92/94/95/96/98
17120:
17121: ****
17122: Procedure
17123: PROCEDURE SIZE 7507 7401 6792 6310 5971 4438 3797 3600 7881 7938
17124: Procedure *****Now the Procedure list*****
17125: Procedure ( ACol, ARow : Integer; Items : TStrings)
17126: Procedure ( Agg : TAggregate)
17127: Procedure ( ASender : TComponent; const AReplyStatus : TReplyStatus)
17128: Procedure ( ASender : TComponent; const AString : String; var AMsg : TIdMessage)
17129: Procedure ( ASender : TComponent; var AMsg : TIdMessage)
17130: Procedure ( ASender : TObject; const ABytes : Integer)
17131: Procedure ( ASender : TObject; VStream : TStream)
17132: Procedure ( AThread : TIdThread)
17133: Procedure ( AWebModule : TComponent)
17134: Procedure ( Column : TColumn)
17135: Procedure ( const AUsername : String; const APASSWORD : String; AAAuthenticationResult : Boolean)
17136: Procedure ( const iStart : integer; const sText : string)
17137: Procedure ( Control : TCustomTabControl; TabIndex : Integer; const Rect : TRect; Active : Boolean)
17138: Procedure ( Database : TDatabase; LoginParams : TStrings)
17139: Procedure ( DataSet:TCustomClientDataSet;E:EReconcileError;UpdateKind:TUpdateKind;var Action:TReconcileAction)
17140: Procedure ( DATASET : TDATASET)
17141: Procedure ( DataSet:TDataSet; E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
17142: Procedure ( DATASET : TDATASET; E : TObject; var ACTION : TDATAACTION)
17143: Procedure ( DataSet : TDataSet; UpdateKind : TUpdateKind; var UpdateAction : TUpdateAction)
17144: Procedure ( DATASET : TDATASET; var ACCEPT : BOOLEAN)
17145: Procedure ( DBCtrlGrid : TDBCtrlGrid; Index : Integer)
17146: Procedure ( Done : Integer)
17147: Procedure ( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
17148: Procedure ( HeaderControl:TCustomHeaderControl;Section:THeaderSection; const Rect:TRect;Pressed:Boolean)
17149: Procedure
  (HeaderControl:TCustomHeaderControl;Section:THeaderSection;Width:Integer;State:TSectionTrackState)
17150: Procedure ( HeaderControl : THeaderControl; Section : THeaderSection)
17151: Procedure ( HeaderControl:THeaderControl;Section: THeaderSection; const Rect:TRect; Pressed : Boolean)
17152: Procedure ( HeaderControl:THeaderControl;Section:THeaderSection;Width:Integer;State : TSectionTrackState)
17153: Procedure ( Sender : TCustomListView;const ARect : TRect;Stage:TCustomDrawStage;var DefaultDraw: Boolean)
17154: Procedure ( Sender : TCustomListView; const ARect : TRect; var DefaultDraw : Boolean)
17155: Procedure ( Sender : TCustomListView; Item : TListItem; Rect : TRect; State : TownerDrawState)
17156: Procedure ( Sender : TCustomTreeView; const ARect : TRect; var DefaultDraw : Boolean)
17157: Procedure ( SENDER : TFIELD; const TEXT : String)
17158: Procedure ( SENDER : TFIELD; var TEXT : STRING; DISPLAYTEXT : BOOLEAN)
17159: Procedure ( Sender : TIdTelnet; const Buffer : String)
17160: Procedure ( Sender : TIdTelnet; Status : TIdTelnetCommand)
17161: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; SELECTED : BOOLEAN)
17162: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
17163: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; var WIDTH, HEIGHT : INTEGER)
17164: Procedure ( Sender : TObject; ACol, ARow : Longint; const Value : string)
17165: Procedure ( Sender : TObject; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
17166: Procedure ( Sender : TObject; ACol, ARow : Longint; var CanSelect : Boolean)
17167: Procedure ( Sender : TObject; ACol, ARow : Longint; var Value : string)
17168: Procedure ( Sender : TObject; Button : TMPBtnType)
17169: Procedure ( Sender : TObject; Button : TMPBtnType; var DoDefault : Boolean)
17170: Procedure ( Sender : TObject; Button : TUDBtnType)
17171: Procedure ( Sender : TObject; Canvas : TCanvas; PageRect : TRect; var DoneDrawing : Boolean)
17172: Procedure ( Sender : TObject; ClientSocket : TServerClientWinSocket; var SocketThread : TServerClientThread)
17173: Procedure ( Sender : TObject; Column : TListColumn)
17174: Procedure ( Sender : TObject; Column : TListColumn; Point : TPoint)
17175: Procedure ( Sender : TObject; Connecting : Boolean)
17176: Procedure ( Sender:TObject;const PapSize:SmallInt;const Orient:TPrtOrient;const PageTy:TPageTy;var DoneDraw:Bool
17177: Procedure ( Sender:TObject; const Rect : TRect; DataCol : Integer; Column: TColumn; State : TGridDrawState)
17178: Procedure ( Sender : TObject; const Rect : TRect; Field : TField; State : TGridDrawState)
17179: Procedure ( Sender : TObject; const UserString:string;var DateAndTime:TDate;var AllowChange:Boolean)
17180: Procedure ( Sender : TObject; E : Exception; var Handled : Boolean)
17181: Procedure ( Sender : TObject; FromIndex, ToIndex : Longint)
17182: Procedure ( Sender : TObject; FromSection, ToSection : THeaderSection; var AllowDrag : Boolean)
17183: Procedure ( Sender : TObject; Index : LongInt)
17184: Procedure ( Sender : TObject; Item : TListItem)
17185: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange)
17186: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange; var AllowChange : Boolean)
17187: Procedure ( Sender : TObject; Item : TListItem; Selected : Boolean)
17188: Procedure ( Sender : TObject; Item : TListItem; var AllowEdit : Boolean)
17189: Procedure ( Sender : TObject; Item : TListItem; var S : string)
17190: Procedure ( Sender : TObject; Item1, Item2 : TListItem; Data : Integer; var Compare : Integer)
17191: Procedure ( Sender : TObject; ModalResult : TModalResult; var CanClose : Boolean)
17192: Procedure ( Sender : TObject; Month : LongWord; var MonthBoldInfo : LongWord)
17193: Procedure ( Sender : TObject; NewTab : Integer; var AllowChange : Boolean)

```

```

17194: Procedure ( Sender : TObject; Node : TTreenode)
17195: Procedure ( Sender : TObject; Node : TTreenode; var AllowChange : Boolean)
17196: Procedure ( Sender : TObject; Node : TTreenode; var AllowCollapse : Boolean)
17197: Procedure ( Sender : TObject; Node : TTreenode; var AllowEdit : Boolean)
17198: Procedure ( Sender : TObject; Node : TTreenode; var AllowExpansion : Boolean)
17199: Procedure ( Sender : TObject; Node : TTreenode; var S : string)
17200: Procedure ( Sender : TObject; Node1, Node2 : TTreenode; Data : Integer; var Compare : Integer)
17201: Procedure ( Sender : TObject; NumObjects, NumChars : Integer; var SaveClipboard : Boolean)
17202: Procedure ( Sender : TObject; Rect : TRect)
17203: Procedure ( Sender : TObject; Request : TWebRequest; Response : TWebResponse; var Handled : Boolean)
17204: Procedure ( Sender : TObject; Shift : TShiftState; X, Y : Integer; Orient : TPageScrollerOrientation; var Delta : Int)
17205: Procedure ( Sender : TObject; Socket : TCustomWinSocket)
17206: Procedure ( Sender : TObject; Socket : TCustomWinSocket; ErrorEvent : TErrorEvent; var ErrorCode : Integer)
17207: Procedure ( Sender : TObject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
17208: Procedure ( Sender : TObject; Socket : TSocket; var ClientSocket : TServerClientWinSocket)
17209: Procedure ( Sender : TObject; SOURCE : TMENUTITEM; REBUILD : BOOLEAN)
17210: Procedure ( Sender : TObject; StartPos, EndPos : Integer; var AllowChange : Boolean)
17211: Procedure ( Sender : TObject; TabCanvas : TCanvas; R : TRect; Index : Integer; Selected : Boolean)
17212: Procedure ( Sender : TObject; TabIndex : Integer; var ImageIndex : Integer)
17213: Procedure ( Sender : TObject; Thread : TServerClientThread)
17214: Procedure ( Sender : TObject; TickCount : Cardinal; var Reset : Boolean)
17215: Procedure ( Sender : TObject; Username, Password : string)
17216: Procedure ( Sender : TObject; var AllowChange : Boolean)
17217: Procedure ( Sender : TObject; var AllowChange : Boolean; NewValue : SmallInt; Direction : TUpDownDirection)
17218: Procedure ( Sender : TObject; var Caption : string; var Alignment : THMLCaptionAlignment)
17219: Procedure ( Sender : TObject; var Continue : Boolean)
17220: Procedure ( Sender : TObject; var dest : string; var NumRedirect : Int; var Handled : bool; var VMethod : TIdHTTPMethod)
17221: Procedure ( Sender : TObject; var Username : string)
17222: Procedure ( Sender : TObject; Wnd : HWND)
17223: Procedure ( Sender : TToolbar; Button : TToolButton)
17224: Procedure ( Sender : TToolbar; const ARect : TRect; Stage : TCustomDrawStage; var DefaultDraw : Boolean)
17225: Procedure ( Sender : TToolBar; const ARect : TRect; var DefaultDraw : Boolean)
17226: Procedure ( Sender : TToolBar; Index : Integer; var Allow : Boolean)
17227: Procedure ( Sender : TToolBar; Index : Integer; var Button : TToolButton)
17228: Procedure ( StatusBar : TCustomStatusbar; Panel : TStatusPanel; const Rect : TRect)
17229: Procedure ( StatusBar : TStatusbar; Panel : TStatusPanel; const Rect : TRect)
17230: Procedure ( var FieldNames : TWideStrings; SQL : WideString; var BindAllFields : Boolean; var TableName : WideString)
17231: Procedure ( var FieldNames : TWideStrings; SQL : WideString; var TableName : WideString)
17232: procedure ( Sender : TObject)
17233: procedure ( Sender : TObject; var Done : Boolean)
17234: procedure ( Sender : TObject; var Key : Word; Shift : TShiftState);
17235: procedure _T(Name : tbtString; v : Variant);
17236: Procedure AbandonSignalHandler( RtlSigNum : Integer)
17237: Procedure Abort
17238: Procedure About1Click( Sender : TObject)
17239: Procedure Accept( Socket : TSocket)
17240: Procedure AESSymmetricExecute( const plaintext, ciphertext, password : string)
17241: Procedure AESEncryptFile( const plaintext, ciphertext, password : string)
17242: Procedure AESDecryptFile( const replaintext, ciphertext, password : string)
17243: Procedure AESEncryptString( const plaintext : string; var ciphertext : string; password : string)
17244: Procedure AESDecryptString( var plaintext : string; const ciphertext : string; password : string)
17245: Procedure Add( Addend1, Addend2 : TMyBigInt)
17246: Procedure ADD( const AKEY, AVALUE : VARIANT)
17247: Procedure Add( const Key : string; Value : Integer)
17248: Procedure ADD( const NAME, FIELDS : String; OPTIONS : TINDEXOPTIONS)
17249: Procedure ADD( FIELD : TFIELD)
17250: Procedure ADD( ITEM : TMENUTITEM)
17251: Procedure ADD( POPUP : TPOPUPMENU)
17252: Procedure AddCharacters( xCharacters : TCharSet)
17253: Procedure AddDriver( const Name : string; List : TStrings)
17254: Procedure AddImages( Value : TCustomImageList)
17255: Procedure AddIndex( const Name, Fields : string; Options : TIndexOptions; const DescFields : string)
17256: Procedure AddLambdaTransitionTo( oState : TniRegularExpressionState)
17257: Procedure AddLoader( Loader : TBitmapLoader)
17258: Procedure ADDPARAM( VALUE : TPARAM)
17259: Procedure AddPassword( const Password : string)
17260: Procedure AddStandardAlias( const Name, Path, DefaultDriver : string)
17261: Procedure AddState( oState : TniRegularExpressionState)
17262: Procedure AddStrings( Strings : TStrings);
17263: procedure AddStrings( Strings : TStrings);
17264: Procedure AddStrings1( Strings : TWideStrings);
17265: Procedure AddStringTerm( var sString : string; const sTerm : string; const sSeparator : string)
17266: Procedure AddToRecentDocs( const Filename : string)
17267: Procedure AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset)
17268: Procedure AllFunctionsList1Click( Sender : TObject)
17269: procedure AllObjectsList1Click( Sender : TObject);
17270: Procedure Allocate( AAllocateBytes : Integer)
17271: procedure AllResourceList1Click( Sender : TObject);
17272: Procedure AnsiAppend( var dst : AnsiString; const src : AnsiString)
17273: Procedure AnsiAssign( var dst : AnsiString; var src : AnsiString)
17274: Procedure AnsiDelete( var dst : AnsiString; index, count : Integer)
17275: Procedure AnsiFree( var s : AnsiString)
17276: Procedure AnsiFromWide( var dst : AnsiString; const src : WideString)
17277: Procedure AnsiInsert( var dst : AnsiString; const src : AnsiString; index : Integer)
17278: Procedure AnsiSetLength( var dst : AnsiString; len : Integer)
17279: Procedure AnsiString_to_stream( const Value : ansistring; Destin : TStream)
17280: Procedure AntiFreeze;
17281: Procedure APPEND
17282: Procedure Append( const S : WideString)

```

```

17283: procedure Append(S: string);
17284: Procedure AppendByte( var VBytes : TIdBytes; AByte : byte)
17285: Procedure AppendBytes( var VBytes : TIdBytes; AAdd : TIdBytes)
17286: Procedure AppendChunk( Val : OleVariant)
17287: Procedure AppendData( const Data : OleVariant; HitEOF : Boolean)
17288: Procedure AppendStr( var Dest : string; S : string)
17289: Procedure AppendString( var VBytes : TIdBytes; const AStr : String; ALength : Integer)
17290: Procedure ApplyRange
17291: procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
17292: Procedure Arrange( Code : TListArrangement)
17293: procedure Assert(expr : Boolean; const msg: string);
17294: procedure Assert2(expr : Boolean; const msg: string);
17295: Procedure Assign( Alist : TCustomBucketList)
17296: Procedure Assign( Other : TObject)
17297: Procedure Assign( Source : TDragObject)
17298: Procedure Assign( Source : TPersistent)
17299: Procedure Assign(Source: TPersistent)
17300: procedure Assign2(mystring, mypath: string);
17301: Procedure AssignCurValues( Source : TDataSet);
17302: Procedure AssignCurValues1( const CurValues : Variant);
17303: Procedure ASSIGNFIELD( FIELD : TFIELD)
17304: Procedure ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT)
17305: Procedure AssignFile(var F: Text; FileName: string)
17306: procedure AssignFile(var F: TextFile; FileName: string)
17307: procedure AssignFileRead(var mystring, myfilename: string);
17308: procedure AssignFileWrite(mystring, myfilename: string);
17309: Procedure AssignTo( Other : TObject)
17310: Procedure AssignValues( Value : TParameters)
17311: Procedure ASSIGNVALUES( VALUE : TPARAMS)
17312: Procedure AssociateExtension( IconPath, ProgramName, Path, Extension : string)
17313: Procedure Base64_to_stream( const Base64 : ansiString; Destin : TStream)
17314: Procedure Base64ToVar( NatData : Pointer; const SoapData : WideString);
17315: Procedure Base64ToVar1( var V : Variant; const SoapData : WideString);
17316: Procedure BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
17317: Procedure BcdDivide( Dividend, Divisor : string; var bcdOut : TBcd);
17318: Procedure BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
17319: Procedure BcdDivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
17320: Procedure BcdDivide3( const Dividend : TBcd; const Divisor : string; var bcdOut : TBcd);
17321: Procedure BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
17322: Procedure BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
17323: Procedure BcdMultiply2( const bcdIn : TBcd; const StringIn : string; var bcdOut : TBcd);
17324: Procedure BcdMultiply3( StringIn1, StringIn2 : string; var bcdOut : TBcd);
17325: Procedure BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
17326: Procedure BcdToBytes( Value : TBcd; Bytes : array of byte)
17327: procedure Beep
17328: Procedure BeepOk
17329: Procedure BeepQuestion
17330: Procedure BeepHand
17331: Procedure BeepExclamation
17332: Procedure BeepAsterisk
17333: Procedure BeepInformation
17334: procedure BEGINDRAG(IMMEDIATE:BOOLEAN)
17335: Procedure BeginLayout
17336: Procedure BeginTimer( const Delay, Resolution : Cardinal)
17337: Procedure BeginUpdate
17338: procedure BeginUpdate;
17339: procedure BigScreen1Click(Sender: TObject);
17340: procedure BinToHex(Buffer: PChar; Text: PChar; BufSize: Integer);
17341: Procedure BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits : Boolean);
17342: Procedure BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits : Boolean);
17343: Procedure BitsToBooleans2( const Bits : Integer; var B : TBooleanArray; AllBits : Boolean);
17344: Procedure BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits : Boolean);
17345: Procedure BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord)
17346: Procedure BooleansToBits( var Dest : Byte; const B : TBooleanArray);
17347: Procedure BooleansToBits1( var Dest : Word; const B : TBooleanArray);
17348: Procedure BooleansToBits2( var Dest : Integer; const B : TBooleanArray);
17349: Procedure BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
17350: Procedure BreakPointMenuClick( Sender : TObject)
17351: procedure BRINGTOFRONT
17352: procedure BringToFront;
17353: Procedure btnBackClick( Sender : TObject)
17354: Procedure btnBrowseClick( Sender : TObject)
17355: Procedure BtnClick( Index : TNavigateBtn)
17356: Procedure btnLargeIconsClick( Sender : TObject)
17357: Procedure BuildAndSendRequest( AURI : TIdURI)
17358: Procedure BuildCache
17359: Procedure BurnMemory( var Buff, BuffLen : integer)
17360: Procedure BurnMemoryStream( Destructo : TMemoryStream)
17361: Procedure CalculateFirstSet
17362: Procedure Cancel
17363: procedure CancelDrag;
17364: Procedure CancelEdit
17365: procedure CANCELHINT
17366: Procedure CancelRange
17367: Procedure CancelUpdates
17368: Procedure CancelWriteBuffer
17369: Procedure Capture1(ADest:TStream;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool;
17370: Procedure Capture2( ADest : TStrings; const ADelim : string; const AIsRFCMessage : Boolean);
17371: Procedure Capture3(ADest:TStrings;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool

```

```

17372: procedure CaptureScreenFormat(vname: string; vextension: string);
17373: procedure CaptureScreenPNG(vname: string);
17374: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
17375: procedure CASCADE;
17376: Procedure CastNativeToSoap(Info: PTypeInfo; var SoapData: WideString; NatData: Pointer; var IsNull: Boolean);
17377: Procedure CastSoapToVariant( SoapInfo : PTypeInfo; const SoapData : WideString; NatData : Pointer );
17378: Procedure cbPathClick( Sender : TObject );
17379: Procedure cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState );
17380: Procedure cedebugAfterExecute( Sender : TPSScript );
17381: Procedure cedebugBreakpoint( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal );
17382: Procedure cedebugCompile( Sender : TPSScript );
17383: Procedure cedebugExecute( Sender : TPSScript );
17384: Procedure cedebugIdle( Sender : TObject );
17385: Procedure cedebugLineInfo( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal );
17386: Procedure CenterHeight( const pc, pcParent : TControl );
17387: Procedure CenterDlg(AForm: TForm; MForm: TForm); { Zentriert Forms }
17388: Procedure CenterForm(AForm: TForm; MForm: TForm); { Zentriert Forms }
17389: Procedure Change;
17390: procedure ChangeBiDiModeAlignment(var Alignment: TAlignment);
17391: Procedure Changed;
17392: Procedure ChangeDir( const ADirName : string );
17393: Procedure ChangeDirUp;
17394: Procedure ChangeEntryTransitions( oNewState : TniRegularExpressionState );
17395: Procedure ChangeLevelBy( Value : TChangeRange );
17396: Procedure ChDir(const s: string);
17397: Procedure Check(Status: Integer);
17398: Procedure CheckCommonControl( CC : Integer );
17399: Procedure CHECKFIELDNAME( const FIELDNAME : String );
17400: Procedure CHECKFIELDNAMES( const FIELDNAMES : String );
17401: Procedure CheckForDisconnect( const ARaiseExceptionIfDisconnected: boolean; const AIgnoreBuffer: bool );
17402: Procedure CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected : Boolean );
17403: Procedure CheckToken( T : Char );
17404: procedure CheckToken(t:char);
17405: Procedure CheckTokenSymbol( const S : string );
17406: procedure CheckTokenSymbol(s:string);
17407: Procedure CheckToolMenuDropdown( ToolButton : TToolButton );
17408: procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
17409: Procedure CIED65ToCIED50( var X, Y, Z : Extended );
17410: Procedure CIELABToBGR( const Source, Target : Pointer; const Count : Cardinal );
17411: procedure CipherFile1Click(Sender: TObject);
17412: Procedure Clear;
17413: Procedure Clear1Click( Sender : TObject );
17414: Procedure ClearColor( Color : TColor );
17415: Procedure CLEARITEM( AITEM : TMENUITEM );
17416: Procedure ClearMapping;
17417: Procedure ClearSelection( KeepPrimary : Boolean );
17418: Procedure ClearWriteBuffer;
17419: Procedure Click;
17420: Procedure Close;
17421: Procedure Close1Click( Sender : TObject );
17422: Procedure CloseDatabase( Database : TDatabase );
17423: Procedure CloseDataSets;
17424: Procedure CloseDialog;
17425: Procedure CloseFile(var F: Text);
17426: Procedure Closure;
17427: Procedure CMYKToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal );
17428: Procedure CMYKToBGR1( const C, M, Y, K, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal );
17429: Procedure CodeCompletionList1Click( Sender : TObject );
17430: Procedure ColEnter;
17431: Procedure Collapse;
17432: Procedure Collapse( Recurse : Boolean );
17433: Procedure ColorRGBToHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word );
17434: Procedure CommaSeparatedToStringList( AList : TStringList; const Value : string );
17435: Procedure CommitFreeAndNil( var Transaction : TDBXTransaction );
17436: Procedure Compile1Click( Sender : TObject );
17437: procedure ComponentCount1Click(Sender: TObject);
17438: Procedure Compress(azipfolder, azipfile: string);
17439: Procedure DeCompress(azipfolder, azipfile: string);
17440: Procedure XZip(azipfolder, azipfile: string);
17441: Procedure XUnZip(azipfolder, azipfile: string);
17442: Procedure Connect( const ATimeout : Integer );
17443: Procedure Connect( Socket : TSocket );
17444: procedure Console1Click(Sender: TObject);
17445: Procedure Continue;
17446: Procedure ContinueCount( var Counter : TJclCounter );
17447: procedure CONTROLDESTROYED(CONTROL:TCONTROL);
17448: Procedure ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : integer );
17449: Procedure ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : integer );
17450: Procedure ConvertImage(vsourc, vddestination: string);
17451: // Ex. ConvertImage(Exepath+'my233.bmp.bmp',Exepath+'mypng111.png')
17452: Procedure ConvertBitmap(vsourc, vddestination: string);
17453: Procedure ConvertToGray(Cnv: TCanvas);
17454: Procedure Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Integer; Length : Integer );
17455: Procedure Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Integer; Count : Integer );
17456: Procedure Copy1( Source : TBytes; Offset : Integer; Buffer : TValueBuffer; Count : Integer );
17457: Procedure CopyBytesToHostLongWord(const ASource:TIdBytes;const ASourceIndex:Integer;var VDest:LongWord);
17458: Procedure CopyBytesToHostWord( const ASource : TIdBytes; const ASourceIndex : Integer; var VDest : Word );
17459: Procedure CopyFrom( mbCopy : TMyBigInt );
17460: Procedure CopyMemoryStream( Source, Destination : TMemoryStream );

```

```

17461: procedure CopyRect(const Dest: TRect; Canvas: TCanvas;const Source: TRect);
17462: Procedure CopyTidByteArray( const ASource : array of Byte; const ASourceIndex : Integer; var VDest : array
  of Byte; const ADestIndex : Integer; const ALength : Integer)
17463: Procedure CopyTidBytes(const ASrc:TidBytes;const ASrcIdx:Int;var VDest:TidBytes;const ADestIdx:Int;const
  ALen:Int)
17464: Procedure CopyTidCardinal( const ASource : Cardinal; var VDest : TidBytes; const ADestIndex : Integer)
17465: Procedure CopyTidInt64( const ASource : Int64; var VDest : TidBytes; const ADestIndex : Integer)
17466: Procedure CopyTidIPv6Address(const ASource:TidIPv6Address; var VDest: TIdBytes; const ADestIndex : Integer)
17467: Procedure CopyTidLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex : Integer)
17468: Procedure CopyTidNetworkLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex:Integer)
17469: Procedure CopyTidNetworkWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
17470: Procedure CopyTidString(const ASource:String;var VDest:TidBytes;const ADestIndex:Integer;ALength: Integer)
17471: Procedure CopyTidWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
17472: Procedure CopyToClipboard
17473: Procedure CountParts
17474: Procedure CreateDataSet
17475: Procedure CreateEmptyFile( const FileName : string)
17476: Procedure CreateFileFromString( const FileName, Data : string)
17477: Procedure CreateFromDelta( Source : TPacketDataSet)
17478: procedure CREATEHANDLE
17479: Procedure CreatePipeStreams(var InPipe:TInputPipeStream;var
  OutPipe:TOutputPipeStream;SecAttr:PSecurityAttributes; BufSize:Longint);
17480: Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
17481: Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const Environment:PChar)
17482: Procedure CreateTable
17483: Procedure CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString)
17484: procedure CSyntax1Click(Sender: TObject);
17485: Procedure CurrencyToComp( Value : Currency; var Result : Comp)
17486: Procedure CURSORPOSCHANGED
17487: procedure CutFirstDirectory(var S: String)
17488: Procedure DataBaseError(const Message: string)
17489: Procedure DateTimeToString( var Result : string; Format : string; DateTime : TDateTime);
17490: procedure DateTimeToString(var Result: string; const Format: string; DateTime: TDateTime)
17491: Procedure DateTimeToSystemTime( DateTime : TDateTime; var SystemTime : TSystemTime)
17492: procedure DateTimeToSystemTime(const DateTime: TDateTime; var SystemTime: TSystemTime);
17493: Procedure DBIError(errorCode: Integer)
17494: Procedure DebugOutput( const AText : string)
17495: Procedure DebugRun1Click( Sender : TObject)
17496: procedure Dec;
17497: Procedure DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word)
17498: procedure DecodeDate(const DateTime: TDateTime; var Year, Month, Day: Word);
17499: Procedure DecodeDateDay( const AValue : TDateTime; out AYear, ADayOfYear : Word)
17500: Procedure DecodeDateMonthWeek(const AValue:TDateTime;out AYear, AMonth,AWeekOfMonth,ADayOfWeek : Word)
17501: Procedure DecodeDateTime(const AValue:TDateTime;out AYear,AMonth,ADay,AHour,AMin,ASec,AMillSec:Word)
17502: Procedure DecodeDateWeek( const AValue : TDateTime; out AYear, AWeekOfYear, ADayOfWeek : Word)
17503: Procedure DecodeDayOfWeekInMonth(const AValue:TDateTime;out AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word)
17504: Procedure DecodeTime( DateTime : TdateTime; var Hour, Min, Sec, MSec : Word)
17505: procedure DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word);
17506: Procedure Decompile1Click( Sender : TObject)
17507: Procedure DefaultDrawColumnCell(const Rect : TRect; DataCol:Integer;Column:TColumn;State:TGridDrawState)
17508: Procedure DefaultDrawDataCell( const Rect : TRect; Field : TField; State : TGridDrawState)
17509: Procedure DeferLayout
17510: Procedure defFileRead
17511: procedure DEFOCUSCONTROL(CONTROL:TWINCONTROL;REMOVING:BOOLEAN)
17512: Procedure DelayMicroseconds( const MicroSeconds : Integer)
17513: Procedure Delete
17514: Procedure Delete( const Afilename : string)
17515: Procedure Delete( const Index : Integer)
17516: Procedure DELETE( INDEX : INTEGER)
17517: Procedure Delete( Index : LongInt)
17518: Procedure Delete( Node : TTreeNode)
17519: procedure Delete(var s: AnyString; ifrom, icount: Longint);
17520: Procedure DeleteAlias( const Name : string)
17521: Procedure DeleteDriver( const Name : string)
17522: Procedure DeleteIndex( const Name : string)
17523: Procedure DeleteKey( const Section, Ident : String)
17524: Procedure DeleteRecords
17525: Procedure DeleteRecords( AffectRecords : TAffectRecords)
17526: Procedure DeleteString( var pStr : String; const pDelStr : string)
17527: Procedure DeleteTable
17528: procedure DelphiSite1Click(Sender: TObject);
17529: Procedure Deselect
17530: Procedure Deselect( Node : TTreeNode)
17531: procedure DestroyComponents
17532: Procedure DestroyHandle
17533: Procedure Diff( var X : array of Double)
17534: procedure Diff(var X: array of Double);
17535: Procedure DirCreate( const DirectoryName : String)');
17536: procedure DISABLEALIGN
17537: Procedure DisableConstraints
17538: Procedure Disconnect
17539: Procedure Disconnect( Socket : TSocket)
17540: Procedure Dispose
17541: procedure Dispose(P: PChar)
17542: Procedure DivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
17543: Procedure DoKey( Key : TDBCtrlGridKey)
17544: Procedure DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
17545: Procedure DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
17546: Procedure Dormant

```

```

17547: Procedure DoubleToBcd1( const AValue : Double; var bcd : TBcd);
17548: Procedure DoubleToBytes( Value : Double; Bytes : array of byte)
17549: Procedure DoubleToComp( Value : Double; var Result : Comp)
17550: Procedure Draw( Canvas : TCanvas; X, Y, Index : Integer; Enabled : Boolean);
17551: procedure Draw(X, Y: Integer; Graphic: TGraphic);
17552: Procedure Draw1(Canvas:TCanvas; X,Y,
  Index:Int;ADrawingStyle:TDrawingStyle;AImageType:TImageType;Enabled:Bool);
17553: Procedure DrawArrow( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
17554: Procedure DrawCheck( ACanvas : TCanvas; Location : TPoint; Size : Integer; Shadow : Boolean)
17555: Procedure DrawChevron( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
17556: Procedure DrawColumnCell( const Rect : TRect; DataCol : Integer; Column : TColumn; State : TGridDrawState)
17557: procedure DrawFocusRect(const Rect: TRect);
17558: Procedure DrawHDIBToTBitmap( HDIB : THandle; Bitmap : TBitmap)
17559: Procedure DRAWMENUTITEM( MENUTITEM: TMENUTITEM; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
17560: Procedure DrawOverlay(Canvas:TCanvas;X,Y:Integer;ImageIndex:Integer;Overlay:TOverlay;Enabled: Boolean);
17561: Procedure DrawOverlay1(Canvas:TCanvas; X,Y:Int; ImageIndex:Int; Overlay : TOVERLAY; ADrawingStyle :
  TDrawingStyle; AImageType : TImageType; Enabled : Boolean);
17562: procedure drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: integer);
17563: Procedure DrawPolyLine( const Canvas : TCanvas; var Points : TPointArray; const ClipRect : TRect)
17564: Procedure DropConnections
17565: Procedure DropDown
17566: Procedure DumpDescription( oStrings : TStrings)
17567: Procedure DumpStateTable( oStrings : TStrings)
17568: Procedure EDIT
17569: Procedure EditButtonClick
17570: Procedure EditFont1Click( Sender : TObject)
17571: procedure Ellipse(Xl, Yl, X2, Y2: Integer);
17572: Procedure Ellipse1( const Rect : TRect);
17573: Procedure EMMS
17574: Procedure Encode( ADest : TStream)
17575: procedure ENDDRAG(DROP:BOOLEAN)
17576: Procedure EndEdit( Cancel : Boolean)
17577: Procedure EndTimer
17578: Procedure EndUpdate
17579: Procedure EraseSection( const Section : string)
17580: Procedure Error( const Ident : string)
17581: procedure Error(Ident:Integer)
17582: Procedure ErrorFmt( const Ident : string; const Args : array of const)
17583: Procedure ErrorStr( const Message : string)
17584: procedure ErrorStr(Message:String)
17585: Procedure Exchange( Index1, Index2 : Integer)
17586: procedure Exchange(Index1, Index2: Integer);
17587: Procedure Exec( FileName, Parameters, Directory : string)
17588: Procedure ExecProc
17589: Procedure ExecSQL( UpdateKind : TUpdateKind)
17590: Procedure Execute
17591: Procedure Execute( const CommandText : WideString; var Params, OwnerData : OleVariant)
17592: Procedure ExecuteAndWait( FileName : string; Visibility : Integer)
17593: Procedure ExecuteCommand(executeFile, paramstring: string)
17594: Procedure ExecuteShell(executeFile, paramstring: string)
17595: Procedure ShellExecuteAndWait(executeFile, paramstring: string);
17596: Procedure ExitThread(ExitCode: Integer); stdcall;
17597: Procedure ExitProcess(ExitCode: Integer); stdcall;
17598: Procedure Expand( AUserName : String; AResults : TStrings)
17599: Procedure Expand( Recurse : Boolean)
17600: Procedure ExportClipboard1Click( Sender : TObject)
17601: Procedure ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress)
17602: Procedure ExtractContentFields( Strings : TStrings)
17603: Procedure ExtractCookieFields( Strings : TStrings)
17604: Procedure ExtractFields( Separators,WhiteSpace : TSysCharSet; Content : PChar; Strings : TStrings)
17605: Procedure ExtractHeaderFields(Separ,
 WhiteSpace:TSysChSet;Cont:PChar;Strings:TStrings;Decode:Bool;StripQuotes:Bool)
17606: Procedure ExtractHTTPFields(Separators,WhiteSpace:
  TSysCharSet;Content:PChar;Strings:TStrings;StripQuotes:Bool)
17607: Procedure ExtractQueryFields( Strings : TStrings)
17608: Procedure FastDegToGrad
17609: Procedure FastDegToRad
17610: Procedure FastGradToDeg
17611: Procedure FastGradToRad
17612: Procedure FastRadToDeg
17613: Procedure FastRadToGrad
17614: Procedure FileClose( Handle : Integer)
17615: Procedure FileClose(handle: integer)
17616: procedure FilesFromWildcard( Dir,Mask:string;var Files:TStringList;Subdirs>ShowDirs,Multitasking:Bool)
17617: Procedure FileStructure( AStructure : TIIDFTPDataStructure)
17618: Procedure FillByte2(var X: Byte ; count: integer; value: byte)
17619: Procedure FillBytes( var VBytes : TIIDBytes; const ACount : Integer; const AValue : Byte)
17620: Procedure FillChar( Buffer : TRecordBuffer; Length : Integer; value : Byte)
17621: Procedure FillChar2(var X: PChar ; count: integer; value: char)
17622: Procedure FillCharS(var p: string; count: integer; value: char); //fix3.8
17623: Procedure FillIPList
17624: procedure FillRect(const Rect: TRect);
17625: Procedure FillTStrings( AStrings : TStrings)
17626: Procedure FilterOnBookmarks( Bookmarks : array of const)
17627: procedure FinalizePackage(Module: HMODULE)
17628: procedure FindClose;
17629: procedure FindClose2(var F: TSearchRec)
17630: Procedure FindMatches( const sString : string; xNotify : TniRegularExpressionMatchFoundEvent);
17631: Procedure FindMatches1(const sString:string;iStart:integer;xNotify:TniRegularExpressionMatchFoundEvent);

```

```

17632: Procedure FindNearest( const KeyValues : array of const )
17633: Procedure FinishContext
17634: Procedure FIRST
17635: Procedure FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float )
17636: Procedure FloatToDecimal(var Result:TFloatRec;const Value:extend;ValueType:TFloatValue;Precis,Decs:Int );
17637: Procedure FloodFill( X, Y : Integer; Color : TColor; FillStyle : TFillStyle )
17638: Procedure FlushSchemaCache( const TableName : string )
17639: procedure FmtStr(var Result: string; const Format: string; const Args: array of const )
17640: procedure FOCUSCONTROL(CONTROL:TWINCONTROL)
17641: Procedure Form1Close( Sender : TObject; var Action : TCloseAction )
17642: Procedure FormActivate( Sender : TObject )
17643: procedure FormatIn(const format: String; const args: array of const ); //alias
17644: Procedure FormClose( Sender : TObject; var Action : TCloseAction )
17645: Procedure FormCreate( Sender : TObject )
17646: Procedure FormDestroy( Sender : TObject )
17647: Procedure FormKeyPress( Sender : TObject; var Key : Char )
17648: procedure FormOutput1Click(Sender: TObject);
17649: Procedure FormToHTML( Form : TForm; Path : string )
17650: procedure FrameRect(const Rect: TRect);
17651: Procedure Frame3D(Canvas:TCanvas; var Rect:TRect; TopColor,BottomColor:TColor; Width : Integer)
17652: Procedure NotebookHandlesNeeded( Notebook : TNotebook )
17653: Procedure Free( Buffer : TRecordBuffer )
17654: Procedure Free( Buffer : TValueBuffer )
17655: Procedure Free;
17656: Procedure FreeAndNil(var Obj:TObject)
17657: Procedure FreeImage
17658: procedure FreeMem(P: PChar; Size: Integer)
17659: Procedure FreeTreeData( Tree : TUpdateTree )
17660: Procedure Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Integer )
17661: Procedure FullCollapse
17662: Procedure FullExpand
17663: Procedure GenerateDPB(sl: TStrings; var DPB: string; var DPBLength: Short); //InterBase
17664: Procedure GenerateTPB(sl: TStrings; var TPB: string; var TPBLength: Short);
17665: Procedure OUTPUTXML( SQLOBJECT : TIBSQL; OUTPUTOBJECT : TIBOUTPUTXML )
17666: Procedure Get1( AURL : string; const AResponseContent : TStream );
17667: Procedure Get1( const ASourceFile : string; ADest : TStream; AResume : Boolean );
17668: Procedure Get2(const ASourceFile,ADestFile: string;const ACAnOverwrite: boolean; AResume: Boolean );
17669: Procedure GetAliasNames( List : TStrings )
17670: Procedure GetaliasParams( const AliasName : string; List : TStrings )
17671: Procedure GetApplicationsRunning( Strings : TStrings )
17672: Procedure getBox(aURL, extension: string);
17673: Procedure GetCommandTypes( List : TWideStrings )
17674: Procedure GetConfigParams( const Path, Section : string; List : TStrings )
17675: Procedure GetConnectionNames( List : TStrings; Driver : string; DesignMode : Boolean )
17676: Procedure GetConvFamilies( out AFamilies : TConvFamilyArray )
17677: Procedure GetConvTypes( const AFamily : TConvFamily; out ATypes : TConvTypeArray )
17678: Procedure GetDatabaseNames( List : TStrings )
17679: Procedure GetDataPacket( DataSet : TDataSet; var RecsOut : Integer; out Data : OleVariant )
17680: Procedure GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize: longWORD; var ImageSize : longWORD )
17681: Procedure GetDir(d: byte; var s: string)
17682: Procedure GetDirList( const Search : string; List : TStrings; Recursive : Boolean )
17683: Procedure GetDriverNames( List : TStrings )
17684: Procedure GetDriverNames( List : TStrings; DesignMode : Boolean )
17685: Procedure GetDriverParams( const DriverName : string; List : TStrings )
17686: Procedure GetEmails1Click( Sender : TObject )
17687: Procedure getEnvironmentInfo;
17688: Function getEnvironmentString: string;
17689: Procedure GetFieldNames( const DatabaseName, TableName : string; List : TStrings )
17690: Procedure GetFieldNames( const TableName : string; List : TStrings )
17691: Procedure GetFieldNames( const TableName : string; List : TStrings );
17692: Procedure GetFieldNames( const TableName : WideString; List : TWideStrings );
17693: Procedure GETFIELDNAMES( LIST : TSTRINGS )
17694: Procedure GetFieldNames1( const TableName : string; List : TStrings );
17695: Procedure GetFieldNames1( const TableName : string; SchemaName : string; List : TStrings );
17696: Procedure GetFieldNames2( const TableName : WideString; SchemaName : WideString; List : TWideStrings );
17697: Procedure GetFieldNames3( const TableName : WideString; List : TWideStrings );
17698: Procedure GetFileAttributeList( const Items : TStrings; const Attr : Integer )
17699: Procedure GetFileAttributeListEx( const Items : TStrings; const Attr : Integer )
17700: Procedure GetFMTBcd( Buffer : TRecordBuffer; var value : TBcd )
17701: Procedure GetFormatSettings
17702: Procedure GetFromDIB( var DIB : TBitmapInfo )
17703: Procedure GetFromHDIB( HDIB : HBitmap )
17704: Procedure GetIcon( Index : Integer; Image : TIcon );
17705: Procedure GetIcon1(Index : Integer; Image:TIcon; ADrawingStyle:TDrawingStyle; AImageType:TImageType );
17706: Procedure GetIndexInfo( IndexName : string )
17707: Procedure GetIndexNames( const TableName, SchemaName : string; List : TStrings );
17708: Procedure GetIndexNames( List : TStrings )
17709: Procedure GetIndexNames1( const TableName : WideString; List : TWideStrings );
17710: Procedure GetIndexNames2( const TableName, SchemaName : WideString; List : TWideStrings );
17711: Procedure GetIndexNames4( const TableName : string; List : TStrings );
17712: Procedure GetInternalResponse
17713: Procedure GETITEMNAMES( LIST : TSTRINGS )
17714: procedure GetMem(P: PChar; Size: Integer)
17715: Procedure GETOLE2ACCELERATORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:INTEGER;GROUPS:array of INTEGER)
17716: procedure GetPackageDescription(ModuleName: PChar): string
17717: Procedure GetPackageName( List : TStrings );
17718: Procedure GetPackageName1( List : TWidestrings );
17719: Procedure GetParamList( List : TList; const ParamNames : WideString )
17720: Procedure GetProcedureNames( List : TStrings );

```

```

17721: Procedure GetProcedureNames( List : TWideStrings );
17722: Procedure GetProcedureNames1( const PackageName : string; List : TStrings );
17723: Procedure GetProcedureNames1( List : TStrings );
17724: Procedure GetProcedureNames2( const PackageName, SchemaName : string; List : TStrings );
17725: Procedure GetProcedureNames3( List : TWideStrings );
17726: Procedure GetProcedureNames4( const PackageName : Widestring; List : TWideStrings );
17727: Procedure GetProcedureNames5( const PackageName, SchemaName : WideString; List : TWideStrings );
17728: Procedure GetProcedureParams( ProcedureName : WideString; List : TList );
17729: Procedure GetProcedureParams1( ProcedureName, PackageName : WideString; List : TList );
17730: Procedure GetProcedureParams2( ProcedureName, PackageName, SchemaName : Widestring; List : TList );
17731: Procedure GetProviderNames( Names : TWideStrings );
17732: Procedure GetProviderNames( Proc : TGetStrProc );
17733: Procedure GetProviderNames1( Names : TStrings );
17734: procedure GetQrCode2(Width,Height:Word; Correct_Level: string; const Data:string; aPath: string);
17735: procedure GetQrCode3(Width,Height:Word;Correct_Level:string;const Data:string;aPath:string); //no open image
17736: Function GetQrCode4(Width,Height:Word;Correct_Level:string; const Data:string;aFormat:string):TLinearBitmap;
17737: Procedure GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte );
17738: Procedure GetSchemaNames( List : TStrings );
17739: Procedure GetSchemaNames1( List : TWideStrings );
17740: Procedure getScriptandRunAsk;
17741: Procedure getScriptandRun(ascript: string);
17742: Procedure getScript(ascript: string); //alias
17743: Procedure getWebScript(ascript: string); //alias
17744: Procedure GetSessionNames( List : TStrings );
17745: Procedure GetStoredProcNames( const DatabaseName : string; List : TStrings );
17746: Procedure GetStrings( List : TStrings );
17747: Procedure GetSystemTime; stdcall;
17748: Procedure GetTableNames(const DatabaseName,Pattern:string;Extensions,SystemTables:Boolean>List:TStrings);
17749: Procedure GetTableNames( List : TStrings; SystemTables : Boolean );
17750: Procedure GetTableNames( List : TStrings; SystemTables : Boolean );
17751: Procedure GetTableNames( List : TWideStrings; SystemTables : Boolean );
17752: Procedure GetTableNames1( List : TStrings; SchemaName : Widestring; SystemTables : Boolean );
17753: Procedure GetTableNames1( List : TStrings; SystemTables : Boolean );
17754: Procedure GetTableNames2( List : TwideStrings; SchemaName : WideString; SystemTables : Boolean );
17755: Procedure GetTransitionsOn( cChar : char; oStateList : TList );
17756: Procedure GetVisibleWindows( List : Tstrings );
17757: Procedure GoBegin
17758: Procedure GotoCurrent( DataSet : TCustomClientDataSet );
17759: Procedure GotoCurrent( Table : TTable );
17760: procedure GotoEnd1Click(Sender: TObject);
17761: Procedure GotoNearest
17762: Procedure GradientFillCanvas(const ACanvas:TCanvas;const AStartCol,AEndCol:TColor;const ARect:TRect;const Direction:TGradientDirection)
17763: Procedure HandleException( E : Exception; var Handled : Boolean )
17764: procedure HANDLEMESSAGE
17765: procedure HandleNeeded;
17766: Procedure Head( AURL : string )
17767: Procedure Help( var AHelpContents : TStringList; ACommand : String )
17768: Procedure HexToBinary( Stream : TStream )
17769: procedure HexToBinary(Stream:TStream)
17770: Procedure HideDragImage
17771: Procedure HideFormCaption( FormHandle : THandle; Hide : Boolean )
17772: Procedure HideTraybar
17773: Procedure HideWindowForSeconds(secs: integer); //3 seconds
17774: Procedure HideWindowForSeconds2(secs: integer; appHandle, aSelf: TForm); //3 seconds
17775: Procedure HookOSExceptions
17776: Procedure HookSignal( RtlSigNum : Integer )
17777: Procedure HSLToRGB( const H, S, L : Single; out R, G, B : Single );
17778: Procedure HTMLSyntax1Click( Sender : TObject )
17779: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler )
17780: Procedure IFPS3ClassesPlugin1ExecImport( Sender : TObject; Exec : TPSEexec; x : TPSRuntimeClassImporter )
17781: Procedure ImportFromClipboard1Click( Sender : TObject )
17782: Procedure ImportFromClipboard2Click( Sender : TObject )
17783: Procedure IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Integer )
17784: procedure Incb(var x: byte);
17785: Procedure IncludelClick( Sender : TObject )
17786: Procedure IncludeOFF; //preprocessing
17787: Procedure IncludeON;
17788: procedure Info1Click(Sender: TObject);
17789: Procedure InitAltRecBuffers( CheckModified : Boolean )
17790: Procedure InitContext( Request : TWebRequest; Response : TWebResponse )
17791: Procedure InitContext(WebModuleList:TAbstractWebModuleList;Request:TWebRequest;Response:TWebResponse)
17792: Procedure InitData( ASource : TDataSet )
17793: Procedure InitDelta( ADelta : TPacketDataSet );
17794: Procedure InitDelta1( const ADelta : OleVariant );
17795: Procedure InitErrorPacket( E : EUpdateError; Response : TResolverResponse )
17796: Procedure Initialize
17797: procedure InitializePackage(Module: HMODULE)
17798: Procedure INITIACTION
17799: Procedure initHexArray(var hexn: THexArray); //THexArray', 'array[0..15] of char;'
17800: Procedure InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet )
17801: Procedure InitModule( AModule : TComponent )
17802: Procedure InitStdConvs
17803: Procedure InitTreeData( Tree : TUpdateTree )
17804: Procedure INSERT
17805: Procedure Insert( Index : Integer; AClass : TClass )
17806: Procedure Insert( Index : Integer; AComponent : TComponent )
17807: Procedure Insert( Index : Integer; AObject : TObject )

```

```

17808: Procedure Insert( Index : Integer; const S : WideString)
17809: Procedure Insert( Index : Integer; Image, Mask : TBitmap)
17810: Procedure Insert(Index: Integer; const S: string);
17811: procedure Insert(Index: Integer; S: string);
17812: procedure Insert(s: AnyString; var s2: AnyString; iPos: Longint);
17813: procedure InsertComponent(AComponent:TComponent)
17814: procedure InsertControl(AControl: TControl);
17815: Procedure InsertIcon( Index : Integer; Image : TIcon)
17816: Procedure InsertMasked( Index : Integer; Image : TBitmap; MaskColor : TColor)
17817: Procedure InsertObject( Index : Integer; const S : WideString; AObject : TObject)
17818: procedure InsertObject(Index:Integer;S:String;AObject:TObject)
17819: Procedure Int16ToBytes( Value : SmallInt; Bytes : array of byte)
17820: Procedure Int32ToBytes( Value : Integer; Bytes : array of byte)
17821: Procedure Int64ToBytes( Value : Int64; Bytes : array of byte)
17822: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
17823: Procedure InternalBeforeResolve( Tree : TUpdateTree)
17824: Procedure InvalidateModuleCache
17825: Procedure InvalidateTitles
17826: Procedure InvalidDateDayError( const AYear, ADayOfYear : Word)
17827: Procedure InvalidDateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
17828: Procedure InvalidDateTimeError(const AYear,AMth,Aday,AMin,ASec,AMilSec:Word;const
  ABaseDate:TDateTime)
17829: Procedure InvalidDateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
17830: Procedure InvalidDayOfWeekInMonthError( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word)
17831: procedure JavaSyntax1Click(Sender: TObject);
17832: Procedure JclLocalesInfoList( const Strings : TStrings; InfoType : Integer)
17833: Procedure KillDataChannel
17834: Procedure Largefont1Click( Sender : TObject)
17835: Procedure LAST
17836: Procedure LaunchCpl( FileName : string)
17837: Procedure Launch( const AFile : string)
17838: Procedure LaunchFile( const AFile : string)
17839: Procedure LetFileList(FileList: TStringlist; apath: string);
17840: Procedure lineToNumber( xmemo : String; met : boolean)
17841: Procedure ListViewCustomDrawItem(Sender:TCustListView;Item:TListItem;State:TCustDrawState;var
  DefaultDraw:Bool)
17842: Procedure ListViewCustomDrawSubItem( Sender : TCustomListView; Item : TListItem; SubItem : Integer; State
  : TCustomDrawState; var DefaultDraw : Boolean)
17843: Procedure ListViewData( Sender : TObject; Item : TListItem)
17844: Procedure ListViewDataFind(Sender:TObject; Find : TItemFind; const FindString : String; const FindPosition
  : TPoint; FindData:Pointer; StartIndex:Integer;Direction:TSearchDirection;Wrap:Boolean;var Index: Integer)
17845: Procedure ListViewDataHint( Sender : TObject; StartIndex, EndIndex : Integer)
17846: Procedure ListViewDblClick( Sender : TObject)
17847: Procedure ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
17848: Procedure ListDLEExports(const FileName: string; List: TStrings);
17849: Procedure Load( const WSDLFileName : WideString; Stream : TMemoryStream)
17850: procedure LoadBytecode1Click(Sender: TObject);
17851: procedure LoadFilefromResource(const FileName: string; ms: TMemoryStream);
17852: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HP)
17853: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
17854: Procedure LoadFromFile( AFileName : string)
17855: Procedure LoadFromFile( const AFileName : string; const AHeadersOnly : Boolean)
17856: Procedure LoadFromFile( const FileName : string)
17857: Procedure LOADFROMFILE( const FILENAME : String; BLOBTYPE : TBLOBTYPE)
17858: Procedure LoadFromFile( const FileName : string; DataType : TDataType)
17859: Procedure LoadFromFile( const FileName : WideString)
17860: Procedure LoadFromFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
17861: Procedure LoadFromFile(const AFileName: string)
17862: procedure LoadFromFile(FileName:string);
17863: procedure LoadFromFile(FileName:String)
17864: Procedure LoadFromResourceID( Instance : THandle; ResID : Integer)
17865: Procedure LoadFromResourceName( Instance : THandle; const ResName : String)
17866: Procedure LoadFromStream( AStream : TStream; const AHeadersOnly : Boolean)
17867: Procedure LoadFromStream( const Stream : TStream)
17868: Procedure LoadFromStream( S : TStream)
17869: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
17870: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
17871: Procedure LoadFromStream( Stream : TStream)
17872: Procedure LOADFROMSTREAM( STREAM : TSTREAM; BLOBTYPE : TBLOBTYPE)
17873: Procedure LoadFromStream( Stream : TStream; DataType : TDataType)
17874: procedure LoadFromStream(Stream: TStream);
17875: Procedure LoadFromStream( Stream : TSeekableStream; const FormatExt : string);
17876: Procedure LoadFromStream2( Stream : TStream; const FormatExt : string);
17877: Procedure LoadFromString( AStrings : TStrings; const MimeSeparator : Char)
17878: Procedure LoadLastFile1Click( Sender : TObject)
17879: { LoadIconToImage loads two icons from resource named NameRes,
  into two image lists ALarge and ASmall}
17880: Procedure LoadIconToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes: string);
17882: Procedure LoadMemo
17883: Procedure LoadParamsFromIniFile( FFileName : WideString)
17884: Procedure Lock
17885: Procedure Login
17886: Procedure MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
17887: Procedure MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
17888: Procedure MakeCaseInsensitive
17889: Procedure MakeDeterministic( var bChanged : boolean)
17890: Procedure MakeGrayPal( var Palette, ColorCount : Integer)
17891: // type TVolumeLevel = 0..127; , saveFilePath as C:\MyFile.wav
17892: Procedure MakeSound(Frequency{Hz}, Duration{mSec}: Int; Volume: TVolumeLevel; savefilePath: string);

```

```

17893: Procedure MakeComplexSound(N:integer;freqlist:TStrings;Duration{mSec}:Int;pinknoise:bool;Volume:Byte);
17894: Procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
17895: Procedure SetRectComplexFormatStr( const S : string)
17896: Procedure SetPolarComplexFormatStr( const S : string)
17897: Procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
17898: Procedure MakeVisible
17899: Procedure MakeVisible( PartialOK : Boolean)
17900: Procedure ManualClick( Sender : TObject)
17901: Procedure MarkReachable
17902: Procedure maxBox; //shows the exe version data in a win box
17903: Procedure MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
17904: Procedure Memo1Change( Sender : TObject)
17905: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Int;var Action:TSynReplaceAction)
17906: Procedure Memo1SpecialLineColors(Sender:TObject;Line:Integer; var Special:Boolean;var FG,BG: TColor)
17907: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
17908: procedure Memory1Click(Sender: TObject);
17909: Procedure MERGE( MENU : TMainMenu)
17910: Procedure MergeChangeLog
17911: procedure MINIMIZE
17912: Procedure MinimizeMaxbox;
17913: Procedure MkDir(const s: string)
17914: Procedure mnuPrintFont1Click( Sender : TObject)
17915: procedure ModalStarted
17916: Procedure Modified
17917: Procedure ModifyAlias( Name : string; List : TStrings)
17918: Procedure ModifyDriver( Name : string; List : TStrings)
17919: Procedure MomentSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4, Skew,Kurtosis:Extended)
17920: Procedure MouseToCell( X, Y : Integer; var ACol, ARow : Longint)
17921: Procedure Move( CurIndex, NewIndex : Integer)
17922: procedure Move(CurIndex, NewIndex: Integer);
17923: procedure Move2(const Source: TByteArray; var Dest: TByteArray; Count: Integer)
17924: Procedure MoveChars(const ASource: String; ASourceStart:int;var ADest: String; ADestStart,ALen:integer)
17925: Procedure moveCube( o : TMyLabel )
17926: Procedure MoveTo( Destination : LongInt; AttachMode : TAttachMode)
17927: procedure MoveTo(X, Y: Integer);
17928: procedure MoveWindowOrg(DC: HDC; DX, DY: Integer);
17929: Procedure MovePoint(var x,y:Extended; const angle:Extended);
17930: Procedure Multiply( Multiplier1, Multiplier2 : TMyBigInt );
17931: Procedure Multiply1( Multiplier1 : TMyBigInt; Multiplier2 : Integer );
17932: Procedure MsgBox(Handle:Int;const Msg,Caption:string;const IcoName:string = 'MAINICON';Flags:DWORD=MB_OK);
17933: Procedure mxButton(x,y,width,height,top,left,ahandle: integer);
17934: Procedure New( Width, Height : Integer; PixFormat : TPixelFormat)
17935: Procedure New(P: PChar)
17936: procedure New1Click(Sender: TObject);
17937: procedure NewInstance1Click(Sender: TObject);
17938: Procedure NEXT
17939: Procedure NextMonth
17940: Procedure Noop
17941: Procedure NormalizePath( var APath : string)
17942: procedure ObjectBinaryToText(Input, Output: TStream)
17943: procedure ObjectBinaryToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
17944: procedure ObjectResourceToText(Input, Output: TStream)
17945: procedure ObjectResourceToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
17946: procedure ObjectTextToBinary(Input, Output: TStream)
17947: procedure ObjectTextToBinary1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
17948: procedure ObjectTextToResource(Input, Output: TStream)
17949: procedure ObjectTextToResource1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
17950: Procedure Open( const Name, Address, Service : string; Port : Word; Block : Boolean)
17951: Procedure Open( const UserID : WideString; const Password : WideString);
17952: Procedure Open;
17953: Procedure open1Click( Sender : TObject)
17954: Procedure OpenCdDrive
17955: Procedure OpenCloseCdDrive( OpenMode : Boolean; Drive : Char)
17956: Procedure OpenCurrent
17957: Procedure OpenFile(vfilenamepath: string)
17958: Procedure OpenDirectory1Click( Sender : TObject)
17959: Procedure OpenDir(adir: string);
17960: Procedure OpenIndexFile( const IndexName : string)
17961: Procedure OpenSchema(const Schema:TSchemaInf;const Restricts:OleVar;const SchemaID:OleVariant;DataSet:TADODataset)
17962: Procedure OpenWriteBuffer( const AThreshhold : Integer)
17963: Procedure OptimizeMem
17964: Procedure Options1( AURL : string);
17965: Procedure OutputDebugString(lpOutputString : PChar)
17966: Procedure PackBuffer
17967: Procedure Paint
17968: Procedure PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch : Boolean)
17969: Procedure PaintToBitmap( Target : TBitmap)
17970: Procedure PaletteChanged
17971: Procedure ParentBidiModeChanged
17972: Procedure PARENTBIDI MODECHANGED( ACONTROL : TOBJECT)
17973: Procedure PasteFromClipboard;
17974: Procedure PasteImage( Source : TLinearBitmap; X, Y : Integer)
17975: Procedure PathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
17976: Procedure PerformEraseBackground(Control: TControl; DC: HDC);
17977: Procedure PError( Text : string)
17978: procedure Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
17979: procedure Pie(X1:Integer;Y1:Integer;X2:Integer;Y2:Integer;X3:Int;Y3:Integer;X4:Integer;Y4:Integer);

```

```

17980: Procedure Play( FromFrame, ToFrame : Word; Count : Integer)
17981: procedure playmp3(mpath: string);
17982: Procedure PlayMP3Click( Sender : TObject)
17983: Procedure PointCopy( var Dest : TPoint; const Source : TPoint)
17984: Procedure PointMove( var P : TPoint; const DeltaX, DeltaY : Integer)
17985: procedure PolyBezier(const Points: array of TPoint);
17986: procedure PolyBezierTo(const Points: array of TPoint);
17987: procedure Polygon(const Points: array of TPoint);
17988: procedure Polyline(const Points: array of TPoint);
17989: Procedure Pop
17990: Procedure POPULATEOLE2MENU(SHAREDMENU: HMENU; GROUPS: array of INT; var WIDTHS : array of LONGINT)
17991: Procedure PopulationVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
17992: Procedure POPUP( X, Y : INTEGER)
17993: Procedure PopupURL(URL : WideString);
17994: Procedure POST
17995: Procedure Post4( AURL : string; const ASource : TStrings; const AResponseContent : TStream);
17996: Procedure Post5( AURL : string; const ASource, AResponseContent : TStream);
17997: Procedure Post6( AURL : string; const ASource : TIdMultiPartFormDataStream; AResponseContent : TStream);
17998: Procedure PostUser( const Email, FirstName, LastName : WideString)
17999: Procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
18000: procedure Pred(X: int64);
18001: Procedure Prepare
18002: Procedure PrepareStatement
18003: Procedure PreProcessXML( AList : TStrings)
18004: Procedure PreventDestruction
18005: Procedure Print( const Caption : string)
18006: procedure PrintBitmap(aGraphic: TGraphic; Title: string);
18007: procedure printf(const format: String; const args: array of const);
18008: Procedure PrintList(Value: TStringList);
18009: Procedure PrintImage(aValue:TBitmap;Style:TBitmapStyle); //TBitmapStyle=(bsNormal,bsCentered,bsStretched)
18010: Procedure Printtout1Click( Sender : TObject)
18011: Procedure ProcessesHeaders
18012: Procedure PROCESSMENUCHAR( var MESSAGE : TWMMENUCHAR)
18013: Procedure ProcessMessage( AMsg : TIdMessage; AHeaderOnly : Boolean);
18014: Procedure ProcessMessage1( AMsg : TIdMessage; const AStream : TStream; AHeaderOnly : Boolean);
18015: Procedure ProcessMessage2( AMsg : TIdMessage; const AFilename : string; AHeaderOnly : Boolean);
18016: Procedure ProcessMessagesOFF; //application.processmessages
18017: Procedure ProcessMessagesON;
18018: Procedure ProcessPath(const EditText:string;var Drive:Char; var DirPart:string; var FilePart : string)
18019: Procedure ProcessPath1(const EditText:string;var Drive:Char; var DirPart:string; var FilePart:string);
18020: Procedure Proclist Size is: 3797 /1415
18021: Procedure procMessClick( Sender : TObject)
18022: Procedure PSScriptCompile( Sender : TPSScript)
18023: Procedure PSScriptExecute( Sender : TPSScript)
18024: Procedure PSScriptLine( Sender : TObject)
18025: Procedure Push( ABoundary : string)
18026: procedure PushItem(AItem: Pointer)
18027: Procedure Put2( AURL : string; const ASource, AResponseContent : TStream);
18028: Procedure Put2( const ASourceFile : string; const ADestFile : string; const AAppend : boolean);
18029: procedure PutLinuxLines(const Value: string)
18030: Procedure Quit
18031: Procedure RaiseConversionError( const AText : string);
18032: Procedure RaiseConversionError1( const AText : string; const AArgs : array of const);
18033: Procedure RaiseConversionRegError( AFamily : TConvFamily; const ADescription : string);
18034: procedure RaiseException(Ex: TIEException; Param: String);
18035: Procedure RaiseExceptionForLastCmdResult;
18036: procedure RaiseLastException;
18037: procedure RaiseException2;
18038: Procedure RaiseLastOSError
18039: Procedure RaiseLastWin32;
18040: procedure RaiseLastWin32Error)
18041: Procedure RaiseListError( const ATemplate : string; const AData : array of const)
18042: Procedure RandomFillStream( Stream : TMemoryStream)
18043: procedure randomize;
18044: Procedure Rasterize( Rasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect)
18045: Procedure RCS
18046: Procedure Read( Socket : TSocket)
18047: Procedure ReadBlobData
18048: procedure ReadBuffer(Buffer:String;Count:LongInt)
18049: procedure Readonly1Click(Sender: TObject);
18050: Procedure ReadSection( const Section : string; Strings : TStrings)
18051: Procedure ReadSections( Strings : TStrings)
18052: Procedure ReadSections( Strings : TStrings);
18053: Procedure ReadSections1( const Section : string; Strings : TStrings);
18054: Procedure ReadSectionValues( const Section : string; Strings : TStrings)
18055: Procedure ReadStream( AStream : TStream; AByteCount:LongInt; const AReadUntilDisconnect : boolean)
18056: Procedure ReadStrings( ADest : TStrings; AReadLinesCount : Integer)
18057: Procedure ReadVersion2(FileName: STRING; aVersion : TStrings);
18058: Function ReadVersion(aFileName: STRING; aVersion : TStrings): boolean;
18059: Procedure Realign;
18060: procedure Rectangle(X1, Y1, X2, Y2: Integer);
18061: Procedure Rectangle1( const Rect : TRect);
18062: Procedure RectCopy( var Dest : TRect; const Source : TRect)
18063: Procedure RectFitToScreen( var R : TRect)
18064: Procedure RectGrow( var R : TRect; const Delta : Integer)
18065: Procedure RectGrowX( var R : TRect; const Delta : Integer)
18066: Procedure RectGrowY( var R : TRect; const Delta : Integer)
18067: Procedure RectMove( var R : TRect; const DeltaX, DeltaY : Integer)
18068: Procedure RectMoveTo( var R : TRect; const X, Y : Integer)

```

```

18069: Procedure RectNormalize( var R : TRect)
18070: // TFileCallbackProcedure = procedure(filename:string);
18071: Procedure RecurseDirectory(Dir: String;IncludeSubs: boolean;callback: TFileCallbackProcedure);
18072: Procedure RecurseDirectory2(Dir: String; IncludeSubs : boolean);
18073: Procedure RedirectTransition(oOldState:TniRegularExpressionState; oNewState : TniRegularExpressionState)
18074: Procedure Refresh;
18075: Procedure RefreshData( Options : TFetchOptions)
18076: Procedure REFRESHLOOKUPLIST
18077: Procedure regExPathfinder(Pathin, fileout, firsttp, aregex, ext: string; asort: boolean);
18078: Procedure RegExPathfinder2(Pathin, fileout, firsttp, aregex, ext: string; asort, acopy: boolean);
18079: Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIAuthenticationClass)
18080: Procedure RegisterChanges( Value : TChangeLink)
18081: Procedure RegisterConversionFormat( const AExtension : string; AConversionClass : TConversionClass)
18082: Procedure RegisterFileFormat( const AExtension, ADescription : string; AGraphicClass : TGraphicClass)
18083: Procedure RegisterFileFormat(Extension,AppID:string;Description:string;Executable:string;IconIndex:Int)
18084: Procedure ReInitialize( ADelay : Cardinal)
18085: procedure RELEASE
18086: Procedure Remove( const AByteCount : integer)
18087: Procedure REMOVE( FIELD : TFIELD)
18088: Procedure REMOVE( ITEM : TMENUITEM)
18089: Procedure REMOVE( POPUP : TPOPUPMENU)
18090: Procedure RemoveAllPasswords
18091: procedure RemoveComponent(AComponent:TComponent)
18092: Procedure RemoveDir( const ADirName : string)
18093: Procedure RemoveLambdaTransitions( var bChanged : boolean)
18094: Procedure REMOVEPARAM( VALUE : TPARAM)
18095: Procedure RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset);
18096: Procedure RemoveTransitionTo1( oState : TniRegularExpressionState);
18097: Procedure Rename( const ASourceFile, ADestFile : string)
18098: Procedure Rename( const FileName : string; Reload : Boolean)
18099: Procedure RenameTable( const NewTableName : string)
18100: Procedure Replace( Index : Integer; Image, Mask : TBitmap)
18101: Procedure Replace1Click( Sender : TObject)
18102: Procedure ReplaceDate( var DateTime : TDateTime; NewDate : TDateTime)
18103: procedure ReplaceDate(var DateTime: TDateTime; const NewDate: TDateTime))
18104: Procedure ReplaceIcon( Index : Integer; Image : TIcon)
18105: Procedure ReplaceMasked( Index : Integer; NewImage : TBitmap; MaskColor : TColor)
18106: Procedure ReplaceTime( var DateTime : TDateTime; NewTime : TDateTime)
18107: procedure ReplaceTime(var DateTime: TDateTime; const NewTime: TDateTime);
18108: Procedure Requery( Options : TExecuteOptions)
18109: Procedure Reset
18110: Procedure Reset1Click( Sender : TObject)
18111: Procedure ResizeCanvas( XSiz, YSiz, XPos, YPos : Integer; Color : TColor)
18112: procedure ResourceExplore1Click(Sender: TObject);
18113: Procedure RestoreContents
18114: Procedure RestoreDefaults
18115: Procedure RestoreOtherInstance( MainFormClassName, MainFormCaption : string)
18116: Procedure RetrieveHeaders
18117: Procedure RevertRecord
18118: Procedure RGBABoBGRA( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal)
18119: Procedure RGBToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18120: Procedure RGBToBGR1( const R, G, B, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
18121: Procedure RGBToHSL( const R, G, B : Single; out H, S, L : Single);
18122: Procedure RGBToHSL1( const RGB : TColor32; out H, S, L : Single);
18123: Procedure RGBToHSV( r, g, b : Integer; var h, s, v : Integer)
18124: Procedure RleCompress2( Stream : TStream)
18125: Procedure RleDecompress2( Stream : TStream)
18126: Procedure RmDir(const S: string)
18127: Procedure Rollback
18128: Procedure Rollback( TransactionDesc : TTransactionDesc)
18129: Procedure RollbackFreeAndNil( var Transaction : TDBXTransaction)
18130: Procedure RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction)
18131: Procedure RollbackTrans
18132: procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer);
18133: Procedure RoundToAllocGranularity64( var Value : Int64; Up : Boolean)
18134: Procedure RoundToAllocGranularityPtr( var Value : Pointer; Up : Boolean)
18135: Procedure RunDll32Internal( Wnd : HWnd; const DLLName, FuncName, CmdLine : string; CmdShow : Integer)
18136: Procedure S_AddMessageToStrings( AMessages : TStrings; AMsg : string)
18137: Procedure S_EBox( const AText : string)
18138: Procedure S_GetEncryptionKeys(DateTime1,DateTime2:TDateTime;var StartKey:int;var MultKey:integer;var AddKey:int)
18139: Procedure S_IBox( const AText : string)
18140: Procedure S_ReplaceChar( var cStr : string; cOldChr, cNewChr : char)
18141: Procedure S_ReplaceStringInFile( AFileName : string; ASearchString, AReplaceString : string)
18142: Procedure S_TokenInit( cBuffer : PChar; const cDelimiters : string)
18143: Procedure SampleVarianceAndMean
18144: ( const X : TDynFloatArray; var Variance, Mean : Float)
18145: Procedure Save2Click( Sender : TObject)
18146: Procedure Saveas3Click( Sender : TObject)
18147: Procedure Savebefore1Click( Sender : TObject)
18148: Procedure SaveBytesToFile(const Data: TBytes; const FileName: string);
18149: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
18150: Procedure SaveConfigFile
18151: Procedure SaveOutput1Click( Sender : TObject)
18152: procedure SaveScreenshotClick(Sender: TObject);
18153: Procedure SaveLn(pathname, content: string); //Saveln(exepath+'mysavelntest.txt', memo2.text);
18154: Procedure SaveToClipboardFormat( var AFormat : Word; var AData : THandle; var APalette : HPALETTE)
18155: Procedure SaveToClipboardFormat( var Format : Word; var Data : THandle; var APalette : HPALETTE)
18156: Procedure SaveToFile( AFileName : string)

```

```

18157: Procedure SAVETOFILE( const FILENAME : String)
18158: Procedure SaveToFile( const FileName : WideString)
18159: Procedure SaveToFile( const FileName : WideString; Format : TPersistFormat)
18160: Procedure SaveToFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
18161: procedure SaveToFile(FileName: string);
18162: procedure SaveToFile(FileName:String)
18163: Procedure SaveToStream( AStream : TStream; const AHeadersOnly : Boolean)
18164: Procedure SaveToStream( OutStream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
18165: Procedure SaveToStream( S : TStream)
18166: Procedure SaveToStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
18167: Procedure SaveToStream( Stream : TStream)
18168: Procedure SaveToStream( Stream : TStream; Format : TDataPacketFormat)
18169: procedure SaveToStream(Stream:TStream);
18170: procedure SaveToStream(Stream:TStream)
18171: Procedure SaveToStream1( Stream : TSeekableStream; const FormatExt : string);
18172: Procedure SaveToStream2( Stream : TStream; const FormatExt : string);
18173: Procedure SaveToStrings( AStrings : TStrings; const MimeSeparator : Char)
18174: procedure Say(const sText: string)
18175: Procedure SBytecodeClick( Sender : TObject)
18176: Procedure ScaleImage( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double)
18177: procedure ScriptExplorer1Click(Sender: TObject);
18178: Procedure Scroll( Distance : Integer)
18179: Procedure Scroll( DX, DY : Integer)
18180: procedure ScrollBy(DeltaX, DeltaY: Integer);
18181: procedure SCROLLINVIEW(ACONTROL:TCONTROL)
18182: Procedure ScrollTabs( Delta : Integer)
18183: Procedure Search1Click( Sender : TObject)
18184: procedure SearchAndOpenDoc(vfilenamepath: string)
18185: procedure SearchAndOpenFile(vfilenamepath: string)
18186: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string)
18187: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
18188: Procedure SearchNext1Click( Sender : TObject)
18189: Procedure Select( Node : TTreeNode; ShiftState : TShiftState);
18190: Procedure Select1( const Nodes : array of TTreeNode);
18191: Procedure Select2( Nodes : Tlist);
18192: Procedure SelectNext( Direction : Boolean)
18193: Procedure SelectNextPage( GoForward : Boolean; CheckTabVisible : Boolean)
18194: Procedure SelfTestPEM //unit uPSI_cPEM
18195: Procedure Send( AMsg : TIdMessage)
18196: //config first in const MAILINIFILE = 'maildef.ini';
18197: //ex.: SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox','
18198: Procedure SendEmail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
18199: Procedure SendMail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
18200: Procedure SendMsg(AMsg : TIdMessage; const AHeadersOnly : Boolean)
18201: Procedure SendMsg(AMsg: TIdMessage; const AHeadersOnly: Boolean = False)
18202: Procedure SendResponse
18203: Procedure SendStream( AStream : TStream)
18204: Procedure Set8087CW( NewCW : Word)
18205: Procedure SetAll( One, Two, Three, Four : Byte)
18206: Procedure SetAltRecBuffers( Old, New, Cur : PChar)
18207: Procedure SetAppDispatcher( const AD dispatcher : TComponent)
18208: procedure SetArrayLength;
18209: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer); //2 dimension
18210: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
18211: Procedure SetAsHandle( Format : Word; Value : THandle)
18212: procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer)
18213: procedure SetCaptureControl(Control: TControl);
18214: Procedure SetColumnAttributes
18215: Procedure SetCookieField(Values:TStrings;const ADomain,APath:string;AExpires:TDateTime;ASecure:Boolean)
18216: Procedure SetCustomHeader( const Name, Value : string)
18217: Procedure SetExprParams(const Text:Widestring;Opts:TFilterOpts;ParserOpts:TParserOpts;const
   FieldName:Widestring)
18218: Procedure SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
18219: Procedure SetFocus
18220: procedure SetFocus; virtual;
18221: Procedure SetInitialState
18222: Procedure SetKey
18223: procedure SetLastError(ErrorCode: Integer)
18224: procedure SetLength;
18225: Procedure SetLineBreakStyle( var T : Text; Style : TTextLineBreakStyle)
18226: Procedure SETOLE2MENUHANDLE( HANDLE : HMENU)
18227: Procedure SetParams( ADataSet : TDataSet; UpdateKind : TUpdateKind);
18228: procedure SETPARAMS(APosition,AMin,AMax:INTEGER)
18229: Procedure SetParams1( UpdateKind : TUpdateKind);
18230: Procedure SetPassword( const Password : string)
18231: Procedure SetPointer( Ptr : Pointer; Size : Longint)
18232: Procedure SetPrimalityTest( const Method : TPrimalityTestMethod)
18233: Procedure SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
18234: Procedure SetProvider( Provider : TComponent)
18235: Procedure SetProxy( const Proxy : string)
18236: Procedure SetPSResult( var PSResult : TPSResult; Value : TObject)
18237: Procedure SetRange( const StartValues, EndValues : array of const)
18238: Procedure SetRangeEnd
18239: Procedure SetRate( const aPercent, aYear : integer)
18240: procedure SetRate(const aPercent, aYear: integer)
18241: Procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
18242: Procedure SetSafeCallExceptionMsg( Msg : String)
18243: procedure SETSELTEXTBUF(BUFFER:PCHAR)
18244: Procedure SetSize( AWidth, AHeight : Integer)

```

```

18245: procedure SetSize(NewSize:LongInt)
18246: procedure SetString(var s: string; buffer: PChar; len: Integer)
18247: Procedure SetStrings( List : TStrings)
18248: Procedure SetText( Text : PwideChar)
18249: procedure SetText(Text: PChar);
18250: Procedure SetTextBuf( Buffer : PChar)
18251: procedure SETTEXTBUF(BUFFER:PCHAR)
18252: Procedure SetTick( Value : Integer)
18253: Procedure SetTimeout( ATimeOut : Integer)
18254: Procedure SetTraceEvent( Event : TDBXTraceEvent)
18255: Procedure SetUserName( const UserName : string)
18256: Procedure SetWallpaper( Path : string);
18257: procedure ShellStyle1Click(Sender: TObject);
18258: Procedure SHORTCUTTOKEY( SHORTCUT : TSHORCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
18259: Procedure ShowFileProperties( const FileName : string)
18260: Procedure ShowInclude1Click( Sender : TObject)
18261: Procedure ShowInterfaces1Click( Sender : TObject)
18262: Procedure ShowLastException1Click( Sender : TObject)
18263: Procedure ShowMessage( const Msg : string)
18264: Procedure ShowMessageBig(const aText : string);
18265: Procedure ShowMessageBig2(const aText : string; aAutoSize: boolean);
18266: Procedure ShowMessageBig3(const aText : string; fsize: byte; aAutoSize: boolean);
18267: Procedure MsgBig(const aText : string); //alias
18268: procedure showMessage(mytext: string);
18269: Procedure ShowMessageFmt( const Msg : string; Params : array of const)
18270: procedure ShowMessageFmt(const Msg: string; Params: array of const))
18271: Procedure ShowMessagePos( const Msg : string; X, Y : Integer)
18272: procedure ShowMessagePos(const Msg: string; X: Integer; Y: Integer))
18273: Procedure ShowSearchDialog( const Directory : string)
18274: Procedure ShowSpecChars1Click( Sender : TObject)
18275: Procedure ShowBitmap(bmap: TBitmap); //draw in a form!
18276: Procedure ShredFile( const FileName : string; Times : Integer)
18277: procedure Shuffle(v0: TStringList);
18278: Procedure ShuffleList( var List : array of Integer; Count : Integer)
18279: Procedure SimulateKeystroke( Key : byte; Shift : TShiftState)
18280: Procedure SinCos( const Theta : Extended; var Sin, Cos : Extended)
18281: Procedure SinCosE( X : Extended; out Sin, Cos : Extended)
18282: Procedure Site( const ACommand : string)
18283: Procedure SkipEOL
18284: Procedure Sleep( ATime : cardinal)
18285: Procedure Sleep( milliseconds : Cardinal)
18286: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
18287: Procedure Slinenumbers1Click( Sender : TObject)
18288: Procedure Sort
18289: Procedure SortColorArray(ColorArray:TColorArray;L,R:Int;SortType:TColorArraySortType;Reverse:Bool)
18290: procedure Speak(const sText: string) //async like voice
18291: procedure Speak2(const sText: string) //sync
18292: procedure Split(Str: string; SubStr: string; List: TStrings);
18293: Procedure SplitNameValuePair( const Line : string; var Name, Value : string)
18294: Procedure SplitColumns( const AData : String; AStrings : TStrings; const ADelim : String)
18295: Procedure SplitColumnsNoTrim( const AData : String; AStrings : TStrings; const ADelim : String)
18296: Procedure SplitLines( AData : PChar; ADataSize : Integer; AStrings : TStrings)
18297: Procedure SplitString( const AStr, AToken : String; var VLeft, VRight : String)
18298: procedure SQLSyntax1Click(Sender: TObject);
18299: Procedure SRand( Seed : RNG_IntType)
18300: Procedure Start
18301: Procedure StartCount( var Counter : TJclCounter; const Compensate : Boolean)
18302: procedure StartFileFinder3(spath,aext,searchstr: string; arecursiv: boolean; reslist: TStringlist);
18303: Procedure StartTransaction( TransDesc : TTransactionDesc)
18304: Procedure Status( var AStatusList : TStringList)
18305: Procedure StatusBar1DblClick( Sender : TObject)
18306: Procedure StepInto1Click( Sender : TObject)
18307: Procedure StepIt
18308: Procedure StepOut1Click( Sender : TObject)
18309: Procedure Stop
18310: procedure stopmp3;
18311: procedure StartWeb(aurl: string);
18312: Procedure StrToInt( integer; astr: string); //of system
18313: Procedure StrDispose( Str : PChar)
18314: procedure StrDispose(Str: PChar)
18315: Procedure StrReplace(var Str: String; Old, New: String);
18316: Procedure StretchDIBits( DC : THandle; const Dest : TRect; HalftoneStretch : Boolean)
18317: procedure StretchDraw(const Rect: TRect; Graphic: TGraphic);
18318: Procedure StringToBytes( Value : String; Bytes : array of byte)
18319: procedure StrSet(c : Char; I : Integer; var s : String);
18320: Procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
18321: Procedure StructureMount( APath : String)
18322: procedure STYLECHANGED(SENDER:TOBJECT)
18323: Procedure Subselect( Node : TTreeNode; Validate : Boolean)
18324: procedure Succ(X: int64);
18325: Procedure SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares : Extended)
18326: procedure SwapChar(var X,Y: char); //swapX follows
18327: Procedure SwapFloats( var X, Y : Float)
18328: procedure SwapGrid(grd: TStringGrid);
18329: Procedure SwapOrd( var I, J : Byte);
18330: Procedure SwapOrd( var X, Y : Integer)
18331: Procedure SwapOrd1( var I, J : Shortint);
18332: Procedure SwapOrd2( var I, J : Smallint);
18333: Procedure SwapOrd3( var I, J : Word);

```

```

18334: Procedure SwapOrd4( var I, J : Integer);
18335: Procedure SwapOrd5( var I, J : Cardinal);
18336: Procedure SwapOrd6( var I, J : Int64);
18337: Procedure SymetricCompareFiles(const plaintext, replaintext: string)
18338: Procedure Synchronize( Method : TMethod);
18339: procedure SyntaxCheck1Click(Sender: TObject);
18340: Procedure SysFreeString(const S: WideString); stdcall;
18341: Procedure TakeOver( Other : TLinearBitmap)
18342: Procedure Talkln(const sText: string) //async voice
18343: procedure tbtn6gresClick(Sender: TObject);
18344: Procedure tbtnUseCaseClick( Sender : TObject);
18345: procedure TerminalStyle1Click(Sender: TObject);
18346: Procedure Terminate
18347: Procedure texSyntax1Click( Sender : TObject)
18348: procedure TextOut(X, Y: Integer; Text: string);
18349: Procedure TextRect( Rect : TRect; X, Y : Integer; const Text : string);
18350: procedure TextRect(Rect: TRect; X: Integer; Y: Integer; const Text: string);
18351: Procedure TextRect1( var Rect : TRect; var Text : string; TextFormat : TTextFormat);
18352: Procedure TextStart
18353: procedure TILE
18354: Procedure TimestampToBytes( Value : TBcd; Bytes : array of byte)
18355: Procedure TitleClick( Column : TColumn)
18356: Procedure ToDo
18357: procedure toolbtnTutorialClick(Sender: TObject);
18358: Procedure Tracel( AURL : string; const AResponseContent : TStream);
18359: Procedure TransferMode( ATransferMode : TIIdFTPTransferMode)
18360: Procedure Truncate
18361: procedure Tutorial101Click(Sender: TObject);
18362: procedure Tutorial10Statistics1Click(Sender: TObject);
18363: procedure Tutorial11Forms1Click(Sender: TObject);
18364: procedure Tutorial11SQL1Click(Sender: TObject);
18365: Procedure tutorial1Click( Sender : TObject)
18366: Procedure tutorial21Click( Sender : TObject)
18367: Procedure tutorial31Click( Sender : TObject)
18368: Procedure tutorial4Click( Sender : TObject)
18369: Procedure Tutorial5Click( Sender : TObject)
18370: procedure Tutorial6Click(Sender: TObject);
18371: procedure Tutorial91Click(Sender: TObject);
18372: Procedure UnhookSignal( RtlSignum: Integer; OnlyIfHooked : Boolean)
18373: procedure UniqueString(var str: AnsiString)
18374: procedure UnLoadLoadPackage(Module: HMODULE)
18375: Procedure Unlock
18376: Procedure UNMERGE( MENU : TMAINMENU)
18377: Procedure UnRegisterChanges( Value : TChangeLink)
18378: Procedure UnregisterConversionFamily( const AFamily : TConvFamily)
18379: Procedure UnregisterConversionType( const AType : TConvType)
18380: Procedure UnRegisterProvider( Prov : TCustomProvider)
18381: Procedure UPDATE
18382: Procedure UpdateBatch( AffectRecords : TAffectRecords)
18383: Procedure UPDATECURSORPOS
18384: Procedure UpdateFile
18385: Procedure UpdateItems( FirstIndex, LastIndex : Integer)
18386: Procedure UpdateResponse( AResponse : TWebResponse)
18387: Procedure UpdateScrollBar
18388: Procedure UpdateView1Click( Sender : TObject)
18389: procedure Val(const s: string; var n, z: Integer)
18390: procedure VarArraySet(c : Variant; I : Integer; var s : Variant);
18391: Procedure VarFMTBcdCreate( var ADest : Variant; const ABcd : TBcd);
18392: Procedure VariantAdd( const src : Variant; var dst : Variant)
18393: Procedure VariantAnd( const src : Variant; var dst : Variant)
18394: Procedure VariantArrayRedim( var V : Variant; High : Integer)
18395: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
18396: Procedure VariantClear( var V : Variant)
18397: Procedure VariantCpy( const src : Variant; var dst : Variant)
18398: Procedure VariantDiv( const src : Variant; var dst : Variant)
18399: Procedure VariantMod( const src : Variant; var dst : Variant)
18400: Procedure VariantMul( const src : Variant; var dst : Variant)
18401: Procedure VariantOr( const src : Variant; var dst : Variant)
18402: Procedure VariantPutElement( var V : Variant; const data : Variant; il : integer);
18403: Procedure VariantPutElement1( var V : Variant; const data : Variant; il, i2 : integer);
18404: Procedure VariantPutElement2( var V : Variant; const data : Variant; il, i2, i3 : integer);
18405: Procedure VariantPutElement3( var V : Variant; const data : Variant; il, i2, i3, i4 : integer);
18406: Procedure VariantPutElement4( var V : Variant; const data : Variant; il, i2, i3, i4, i5 : integer);
18407: Procedure VariantShl( const src : Variant; var dst : Variant)
18408: Procedure VariantShr( const src : Variant; var dst : Variant)
18409: Procedure VariantSub( const src : Variant; var dst : Variant)
18410: Procedure VariantXor( const src : Variant; var dst : Variant)
18411: Procedure VarCastError;
18412: Procedure VarCastError1( const ASourceType, ADestType : TVarType);
18413: Procedure VarInvalidOp
18414: Procedure VarInvalidNullOp
18415: Procedure VarOverflowError( const ASourceType, ADestType : TVarType)
18416: Procedure VarRangeCheckError( const ASourceType, ADestType : TVarType)
18417: Procedure VarArrayCreateError
18418: Procedure VarResultCheck( AResult : HRESULT);
18419: Procedure VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);
18420: Procedure HandleConversionException( const ASourceType, ADestType : TVarType)
18421: Function VarTypeAsText( const AType : TVarType) : string
18422: procedure Voice(const sText: string) //async

```

```

18423: procedure Voice2(const sText: string) //sync
18424: Procedure WaitMiliSeconds( AMSec : word)
18425: Procedure WideAppend( var dst : WideString; const src : WideString)
18426: Procedure WideAssign( var dst : WideString; var src : WideString)
18427: Procedure WideDelete( var dst : WideString; index, count : Integer)
18428: Procedure WideFree( var s : WideString)
18429: Procedure WideFromAnsi( var dst : WideString; const src : AnsiString)
18430: Procedure WideFromPChar( var dst : WideString; src : PChar)
18431: Procedure WideInsert( var dst : WideString; const src : WideString; index : Integer)
18432: Procedure WideSetLength( var dst : WideString; len : Integer)
18433: Procedure WideString2Stream( aWideString : WideString; oStream : TStream)
18434: Procedure WideStringToBytes( Value : WideString; Bytes : array of byte)
18435: Procedure WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
18436: Procedure WinInet_HttpGet(const Url: string; Stream:TStream);
18437: Procedure HttpGet(const Url: string; Stream:TStream);
18438: Procedure WordToTwoBytes( AWord : Word; ByteArray : TIddBytes; Index : integer)
18439: Procedure WordWrap1Click( Sender : TObject)
18440: Procedure Write( const AOut : string)
18441: Procedure Write( Socket : TSocket)
18442: procedure Write(S: string);
18443: Procedure WriteBinaryStream( const Section, Name : string; Value : TStream)
18444: Procedure WriteBool( const Section, Ident : string; Value : Boolean)
18445: Procedure WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow : Boolean)
18446: procedure WriteBuffer(Buffer:String;Count:LongInt)
18447: Procedure WriteCardinal( AValue : Cardinal; const AConvert : Boolean)
18448: Procedure WriteChar( AValue : Char)
18449: Procedure WriteDate( const Section, Name : string; Value : TDateTime)
18450: Procedure WriteDateTime( const Section, Name : string; Value : TDateTime)
18451: Procedure WriteFloat( const Section, Name : string; Value : Double)
18452: Procedure WriteHeader( AHeader : TStrings)
18453: Procedure WriteInteger( AValue : Integer; const AConvert : Boolean)
18454: Procedure WriteInteger( const Section, Ident : string; Value : Longint)
18455: Procedure WriteLine( const AOut : string)
18456: procedure Writeln(s: string);
18457: Procedure WriteLog( const FileName, LogLine : string)
18458: Procedure WriteRFCReply( AReply : TIidRFCReply)
18459: Procedure WriteRFCStrings( AStrings : TStrings)
18460: Procedure WriteSmallInt( AValue : SmallInt; const AConvert : Boolean)
18461: Procedure WriteStream(AStream:TStream;const AAll:Bool;const AWriteByteCount:Bool;const ASIZE:Int)
18462: Procedure WriteString( const Section, Ident, Value : String)
18463: Procedure WriteStrings( AValue : TStrings; const AWriteLinesCount : Boolean)
18464: Procedure WriteTime( const Section, Name : string; Value : TDateTime)
18465: Procedure WriteObjectResourceHeader( ObjStream, Output : TStream)
18466: Procedure Write16bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
18467: Procedure Write32bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
18468: Procedure WriteDataSetToCSV(DataSet: TDataSet; FileName: String)');
18469: procedure WStrSet(c : AnyString; I : Integer; var s : AnyString);
18470: procedure XMLSyntax1Click(Sender: TObject);
18471: Procedure XOR_Streams2( Dest, Srce : TMemoryStream)
18472: Procedure XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
18473: Procedure ZeroFillStream( Stream : TMemoryStream)
18474: procedure XMLSyntax1Click(Sender: TObject);
18475: Procedure ZeroMemory( Ptr : Pointer; Length : Longint)
18476: procedure(Control: TWInControl; Index: Integer; Rect: TRect; State: Byte)
18477: procedure(Control: TWInControl; Index: Integer; var Height: Integer)
18478: procedure(Sender, Source: TObject; X, Y: Integer; State: TDragState; var Accept: Boolean)
18479: procedure(Sender, Source: TObject; X, Y: Integer)
18480: procedure(Sender, Target: TObject; X, Y: Integer)
18481: procedure(Sender: TObject; ASection, AWidth: Integer)
18482: procedure(Sender: TObject; ScrollCode: TScrollCode; var ScrollPos: Integer)
18483: procedure(Sender: TObject; Shift: TShiftState; X, Y: Integer);
18484: procedure(Sender: TObject; var Action: TCcloseAction)
18485: procedure(Sender: TObject; var CanClose: Boolean)
18486: procedure(Sender: TObject; var Key: Char);
18487: ProcedureName ProcedureNames ProcedureParametersCursor @
18488:
18489: *****New Constructors constructor *****
18490: Size is: 1248 1115 996 628 550 544 501 459 (381)
18491: Attach( VersionInfoData : Pointer; Size : Integer)
18492: constructor Create( ABuckets : TBucketListSizes)
18493: Create( ACallBackWnd : HWND)
18494: Create( AClient : TCustomTaskDialog)
18495: Create( AClient : TIidTelnet)
18496: Create( ACollection : TCollection)
18497: Create( ACollection : TFavoriteLinkItems)
18498: Create( ACollection : TTaskDialogButtons)
18499: Create( AConnection : TIidCustomHTTP)
18500: Create( ACreateSuspended : Boolean)
18501: Create( ADataSet : TCustomSQLDataSet)
18502: CREATE( ADATASET : TDATASET)
18503: Create( Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
18504: Create( AGrid : TCustomDBGrid)
18505: Create( AGrid : TStringGrid; AIIndex : Longint)
18506: Create( AHHTTP : TIidCustomHTTP)
18507: Create( AListItems : TlistItems)
18508: Create( AOnBytesRemoved : TIidBufferBytesRemoved)
18509: Create( AOnBytesRemoved : TIidBufferBytesRemoved)
18510: Create( AOwner : TCommonCalendar)
18511: Create( AOwner : TComponent)

```

```

18512: CREATE( AOWNER : TCOMPONENT)
18513: Create( AOwner : TCustomListView)
18514: Create( AOwner : TCustomOutline)
18515: Create( AOwner : TCustomRichEdit)
18516: Create( AOwner : TCustomRichEdit; AttributeType : TAttributeType)
18517: Create( AOwner : TCustomTreeView)
18518: Create( AOwner : TIdUserManager)
18519: Create( AOwner : TListItems)
18520: Create(AOwner:TObj;Handle:hDBICur;CBTyp:CBType;CBBuf:Ptr;CBBufSiz:Int;CallbkEvt:TBDECallbkEvt;Chain:Bool)
18521: CREATE( AOWNER : TPERSISTENT)
18522: Create( AOwner : TPersistent)
18523: Create( AOwner : TTable)
18524: Create( AOwner : TTreeNodes)
18525: Create( AOwner : TWinControl; const ClassName : string)
18526: Create( AParent : TIdCustomHTTP)
18527: Create( AParent : TUpdateTree; AResolver : TCustomResolver)
18528: Create( AProvider : TBaseProvider)
18529: Create( AProvider : TCustomProvider);
18530: Create( AProvider : TDataSetProvider)
18531: Create( ASocket : TCustomWinSocket; TimeOut : Longint)
18532: Create( ASocket : TSocket)
18533: Create( AStrings : TWideStrings)
18534: Create( AToolBar : TToolBar)
18535: Create( ATreeNodes : TTreeNodes)
18536: Create( Autofill : boolean)
18537: Create( AWebPageInfo : TABstractWebPageInfo)
18538: Create( AWebRequest : TWebRequest)
18539: Create( Collection : TCollection)
18540: Create( Collection : TIdMessageParts; ABody : TStrings)
18541: Create( Collection : TIdMessageParts; const AFileName : TFileName)
18542: Create( Column : TColumn)
18543: Create( const AConvFamily : TConvFamily; const ADescription : string)
18544: Create( const AConvFamily : TConvFamily; const ADescription : string; const AFactor : Double)
18545: Create( const AConvFamily : TConvFamily; const ADescription : string; const AToCommonProc,
AFromCommonProc : TConversionProc)
18546: Create( const AInitialState : Boolean; const AManualReset : Boolean)
18547: Create( const ATabSet : TTabSet)
18548: Create( const Compensate : Boolean)
18549: Create( const FileMap : TJclCustomFileMapping; Access, Size : Cardinal; ViewOffset : Int64)
18550: Create( const FileName : string)
18551: Create( const FileName : string; FileMode:Cardinal; const Name:string; Protect:Cardinal; const MaximumSize
: Int64; const SecAttr : PSecurityAttributes)
18552: Create( const FileName : string; FileMode : WordfmShareDenyWrite)
18553: Create( const MaskValue : string)
18554: Create(const Name:string; Protect:Cardinal;const MaximumSize:Int64;const SecAttr:PSecurityAttributes)
18555: Create( const Prefix : string)
18556: Create( const sRegularExpression : string; xFlags : TniRegularExpressionMatchFlags)
18557: Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
18558: Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
18559: Create( CoolBar : TCoolBar)
18560: Create( CreateSuspended : Boolean; ASocket : TServerClientWinSocket)
18561: Create( CreateSuspended : Boolean; ASocket : TServerWinSocket)
18562: Create( DataSet : TDataSet; const
Text:Widestring;Options:TFilterOptions;ParserOptions:TParserOptions;const FieldName : Widestring;
DepFields : TBits; FieldMap : TFieldMap)
18563: Create( DBCtrlGrid : TDBCtrlGrid)
18564: Create( DSTableProducer : TDSTableProducer)
18565: Create( DSTableProducer : TDSTableProducer; ColumnClass : THTMLTableColumnClass)
18566: Create( ErrorCode : DBIResult)
18567: Create( Field : TBlobField; Mode : TBlobStreamMode)
18568: Create( Grid : TCustomDBGrid; ColumnClass : TColumnClass)
18569: Create( HeaderControl : TCustomHeaderControl)
18570: Create( TWebRequest : TWebRequest)
18571: Create( iStart : integer; sText : string)
18572: Create( iValue : Integer)
18573: Create( Kind : TMmTimerKind; Notification : TMmNotificationKind)
18574: Create( MciErrNo : MCIERRO; const Msg : string)
18575: Create(MemoryStream:TCustomMemStream;FreeStream:Boolean;const AIndexOption:TJclMappedTextReaderIndex);
18576: Create( Message : string; ErrorCode : DBResult)
18577: Create( Msg : string)
18578: Create( NativeError, Context : string; ErrCode, PrevError : Integer; E : Exception)
18579: Create( NativeError, Context : string; ErrorCode, PreviousError : DBResult)
18580: Create( oExpression : TniRegularExpression; eType : TniRegularExpressionStateType)
18581: Create( oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags)
18582: Create(oSource:TniRegularExpressionState;oDestination:TniRegularExpressionState;xCharacts:TCharSet;bLambda:bool)
18583: Create( Owner : EDBEngineError; ErrorCode : DBIResult; NativeError : Longint; Message : PChar)
18584: Create( Owner : TCustomComboBoxEx)
18585: CREATE( OWNER : TINDEXDEFS; const NAME, FIELDS: String; OPTIONS: TINDEXOPTIONS)
18586: Create( Owner : TPersistent)
18587: Create( Params : TStrings)
18588: Create( Size : Cardinal)
18589: Create( Socket : TSocket; ServerWinSocket : TServerWinSocket)
18590: Create( StatusBar : TCustomStatusBar)
18591: Create( WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass)
18592: Create(WebResponse : TWebResponse; ItemClass : TCollectionItemClass)
18593: Create(AHandle:Integer)
18594: Create(AOwner: TComponent); virtual;
18595: Create(const AURI : string)
18596: Create(Filename:String;Mode:Word)

```

```

18597: Create(Instance:THandle;ResName:String;ResType:PChar)
18598: Create(Stream : TStream)
18599: CreateDataset(ADataset : TDataset);
18600: CreateDataset(const FileHandle:THandle;const Name:string;Protect:Cardinal;const MaximumSize:Int64;const SecAttr:PSecurityAttributes);
18601: CreateDataset( const FileName : string; const AIIndexOption : TJclMappedTextReaderIndex);
18602: Create2( Other : TObject);
18603: CreateAt(FileMap: TJclCustomFileMapping; Access,Size:Cardinal;ViewOffset:Int64;Address: Pointer)
18604: CreateError(const anErrCode: Integer;const asReplyMessage: string; const asErrorMessage: string)
18605: CreateFmt( MciErrNo : MCIERRO; const Msg : string; const Args : array of const )
18606: CreateFromId(Instance:THandle;ResId:Integer;ResType:PChar)
18607: CreateLinked( DBCtrlGrid : TDBCtrlGrid)
18608: CREATENEW(AOWNER:TCOMPONENT; Dummy: Integer)
18609: CreateRes( Ident : Integer);
18610: CreateRes( MciErrNo : MCIERRO; Ident : Integer)
18611: CreateRes( ResStringRec : PResStringRec);
18612: CreateResHelp( Ident : Integer; AHelpContext : Integer);
18613: CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Integer);
18614: CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
18615: CreateSize( AWidth, AHeight : Integer)
18616: Open( const Name : string; const InheritHandle : Boolean; const DesiredAccess : Cardinal)
18617:
18618: -----
18619: unit UPSI_MathMax;
18620: -----
18621: CONSTS
18622: Bernstein: Float = 0.2801694990238691330364364912307; // Bernstein constant
18623: Cbrt2: Float = 1.2599210498948731647672106072782; // CubeRoot(2)
18624: Cbrt3: Float = 1.4422495703074083823216383107801; // CubeRoot(3)
18625: Cbrt10: Float = 2.1544346900318837217592935665194; // CubeRoot(10)
18626: Cbrt100: Float = 4.6415888336127788924100763509194; // CubeRoot(100)
18627: CbrtPi: Float = 1.4645918875615232630201425272638; // CubeRoot(PI)
18628: Catalan: Float = 0.9159655941772190150546035149324; // Catalan constant
18629: PiJ: Float = 3.1415926535897932384626433832795; // PI
18630: PI: Extended = 3.1415926535897932384626433832795;
18631: PiOn2: Float = 1.5707963267948966192313216916398; // PI / 2
18632: PiOn3: Float = 1.0471975511965977461542144610932; // PI / 3
18633: PiOn4: Float = 0.78539816339744830961566084581988; // PI / 4
18634: Sqrt2: Float = 1.4142135623730950488016887242097; // Sqrt(2)
18635: Sqrt3: Float = 1.7320508075688772935274463415059; // Sqrt(3)
18636: Sqrt5: Float = 2.2360679774997896964091736687313; // Sqrt(5)
18637: Sqrt10: Float = 3.1622776601683793319988935444327; // Sqrt(10)
18638: SqrtPi: Float = 1.7724538509055160272981674833411; // Sqrt(PI)
18639: Sqrt2Pi: Float = 2.506628274631000502415765284811; // Sqrt(2 * PI)
18640: TwoPi: Float = 6.283185307179586476925286766559; // 2 * PI
18641: ThreePi: Float = 9.4247779607693797153879301498385; // 3 * PI
18642: Ln2: Float = 0.69314718055994530941723212145818; // Ln(2)
18643: Ln10: Float = 2.3025850929940456840179914546844; // Ln(10)
18644: LnPi: Float = 1.1447298858494001741434273513531; // Ln(PI)
18645: Log2J: Float = 0.30102999566398119521373889472449; // Log10(2)
18646: Log3: Float = 0.47712125471966243729502790325512; // Log10(3)
18647: LogPi: Float = 0.4971498726941338543512682882909; // Log10(PI)
18648: LogE: Float = 0.43429448190325182765112891891661; // Log10(E)
18649: E: Float = 2.7182818284590452353602874713527; // Natural constant
18650: hLn2Pi: Float = 0.91893853320467274178032973640562; // Ln(2*PI)/2
18651: inv2Pi: Float = 0.15915494309189533576888376337251436203445964574046; // 0.5/Pi
18652: TwoToPower63: Float = 9223372036854775808.0; // 2^63
18653: GoldenMean: Float = 1.618033988749894848204586834365638; // GoldenMean
18654: EulerMascheroni: Float = 0.5772156649015328606065120900824; // Euler GAMMA
18655: RadCor: Double = 57.29577951308232; {number of degrees in a radian}
18656: StDelta : Extended = 0.00001; {delta for difference equations}
18657: StEpsilon : Extended = 0.00001; {epsilon for difference equations}
18658: StMaxIterations : Integer = 100; {max attempts for convergence}
18659:
18660: procedure SIRegister_StdConvs(CL: TPSPPascalCompiler);
18661: begin
18662:   MetersPerInch = 0.0254; // [1]
18663:   MetersPerFoot = MetersPerInch * 12;
18664:   MetersPerYard = MetersPerFoot * 3;
18665:   MetersPerMile = MetersPerFoot * 5280;
18666:   MetersPerNauticalMiles = 1852;
18667:   MetersPerAstronomicalUnit = 1.49598E11; // [4]
18668:   MetersPerLightSecond = 2.99792458E8; // [5]
18669:   MetersPerLightYear = MetersPerLightSecond * 31556925.9747; // [7]
18670:   MetersPerParsec = MetersPerAstronomicalUnit * 206264.806247096; // 60 * 60 * (180 / Pi)
18671:   MetersPerCubit = 0.4572; // [6][7]
18672:   MetersPerFathom = MetersPerFoot * 6;
18673:   MetersPerFurlong = MetersPerYard * 220;
18674:   MetersPerHand = MetersPerInch * 4;
18675:   MetersPerPace = MetersPerInch * 30;
18676:   MetersPerRod = MetersPerFoot * 16.5;
18677:   MetersPerChain = MetersPerRod * 4;
18678:   MetersPerLink = MetersPerChain / 100;
18679:   MetersPerPoint = MetersPerInch * 0.013837; // [7]
18680:   MetersPerPica = MetersPerPoint * 12;
18681:
18682:   SquareMetersPerSquareInch = MetersPerInch * MetersPerInch;
18683:   SquareMetersPerSquareFoot = MetersPerFoot * MetersPerFoot;
18684:   SquareMetersPerSquareYard = MetersPerYard * MetersPerYard;

```

```

18685:     SquareMetersPerSquareMile = MetersPerMile * MetersPerMile;
18686:     SquareMetersPerAcre = SquareMetersPerSquareYard * 4840;
18687:     SquareMetersPerSquareRod = MetersPerRod * MetersPerRod;
18688:
18689:     CubicMetersPerCubicInch = MetersPerInch * MetersPerInch * MetersPerInch;
18690:     CubicMetersPerCubicFoot = MetersPerFoot * MetersPerFoot * MetersPerFoot;
18691:     CubicMetersPerCubicYard = MetersPerYard * MetersPerYard * MetersPerYard;
18692:     CubicMetersPerCubicMile = MetersPerMile * MetersPerMile * MetersPerMile;
18693:     CubicMetersPerAcreFoot = SquareMetersPerAcre * MetersPerFoot;
18694:     CubicMetersPerAcreInch = SquareMetersPerAcre * MetersPerInch;
18695:     CubicMetersPerCord = CubicMetersPerCubicFoot * 128;
18696:     CubicMetersPerCordFoot = CubicMetersPerCubicFoot * 16;
18697:
18698:     CubicMetersPerUSFluidGallon = CubicMetersPerCubicInch * 231; // [2][3][7]
18699:     CubicMetersPerUSFluidQuart = CubicMetersPerUSFluidGallon / 4;
18700:     CubicMetersPerUSFluidPint = CubicMetersPerUSFluidQuart / 2;
18701:     CubicMetersPerUSFluidCup = CubicMetersPerUSFluidPint / 2;
18702:     CubicMetersPerUSFluidGill = CubicMetersPerUSFluidCup / 2;
18703:     CubicMetersPerUSFluidOunce = CubicMetersPerUSFluidCup / 8;
18704:     CubicMetersPerUSFluidTablespoon = CubicMetersPerUSFluidOunce / 2;
18705:     CubicMetersPerUSFluidTeaspoon = CubicMetersPerUSFluidOunce / 6;
18706:
18707:     CubicMetersPerUSDryGallon = CubicMetersPerCubicInch * 268.8025; // [7]
18708:     CubicMetersPerUSDryQuart = CubicMetersPerUSDryGallon / 4;
18709:     CubicMetersPerUSDryPint = CubicMetersPerUSDryQuart / 2;
18710:     CubicMetersPerUSDryPeck = CubicMetersPerUSDryGallon * 2;
18711:     CubicMetersPerUSDryBucket = CubicMetersPerUSDryPeck * 2;
18712:     CubicMetersPerUSDryBushel = CubicMetersPerUSDryBucket * 2;
18713:
18714:     CubicMetersPerUKGallon = 0.00454609; // [2][7]
18715:     CubicMetersPerUKPottle = CubicMetersPerUKGallon / 2;
18716:     CubicMetersPerUKQuart = CubicMetersPerUKPottle / 2;
18717:     CubicMetersPerUKPint = CubicMetersPerUKQuart / 2;
18718:     CubicMetersPerUKGill = CubicMetersPerUKPint / 4;
18719:     CubicMetersPerUKOunce = CubicMetersPerUKPint / 20;
18720:     CubicMetersPerUKPeck = CubicMetersPerUKGallon * 2;
18721:     CubicMetersPerUKBucket = CubicMetersPerUKPeck * 2;
18722:     CubicMetersPerUKBushel = CubicMetersPerUKBucket * 2;
18723:
18724:     GramsPerPound = 453.59237; // [1][7]
18725:     GramsPerDrams = GramsPerPound / 256;
18726:     GramsPerGrains = GramsPerPound / 7000;
18727:     GramsPerTons = GramsPerPound * 2000;
18728:     GramsPerLongTons = GramsPerPound * 2240;
18729:     GramsPerOunces = GramsPerPound / 16;
18730:     GramsPerStones = GramsPerPound * 14;
18731:
18732:     MaxAngle 9223372036854775808.0;
18733:     MaxTanH 5678.261703147071974745965389854);
18734:     MaxFactorial( 1754);
18735:     MaxFloatingPoint(1.189731495357231765085759326628E+4932);
18736:     MinFloatingPoint(3.3621031431120935062626778173218E-4932);
18737:     MaxTanH( 354.89135644669199842162284618659);
18738:     MaxFactorial'LongInt'( 170);
18739:     MaxFloatingPointD(1.797693134862315907729305190789E+308);
18740:     MinFloatingPointD(2.2250738585072013830902327173324E-308);
18741:     MaxTanH( 44.36141955583649980270285577323);
18742:     MaxFactorial'LongInt'( 33);
18743:     MaxFloatingPoints( 3.4028236692093846346337460743177E+38);
18744:     MinFloatingPoints( 1.1754943508222875079687365372222E-38);
18745:     PiExt( 3.1415926535897932384626433832795);
18746:     RatioDegToRad( PiExt / 180.0);
18747:     RatioGradToRad( PiExt / 200.0);
18748:     RatioDegToGrad( 200.0 / 180.0);
18749:     RatioGradToDeg( 180.0 / 200.0);
18750:     Crc16PolynomCCITT'LongWord $1021);
18751:     Crc16PolynomIBM'LongWord $8005);
18752:     Crc16Bits'LongInt'( 16);
18753:     Crc16Bytes'LongInt'( 2);
18754:     Crc16HighBit'LongWord $8000);
18755:     NotCrc16HighBit','LongWord $7FFF);
18756:     Crc32PolynomIEEE','LongWord $04C11DB7);
18757:     Crc32PolynomCastagnoli','LongWord $1EDC6F41);
18758:     Crc32Koopman ','LongWord $741B8CD7);
18759:     Crc32Bits','LongInt'( 32);
18760:     Crc32Bytes','LongInt'( 4);
18761:     Crc32HighBit ','LongWord $80000000);
18762:     NotCrc32HighBit ','LongWord $7FFFFFFF);
18763:
18764:     MinByte      = Low(Byte);
18765:     MaxByte      = High(Byte);
18766:     MinWord      = Low(Word);
18767:     MaxWord      = High(Word);
18768:     MinShortInt  = Low(ShortInt);
18769:     MaxShortInt  = High(ShortInt);
18770:     MinSmallInt  = Low(SmallInt);
18771:     MaxSmallInt  = High(SmallInt);
18772:     MinLongWord   = LongWord(Low(LongWord));
18773:     MaxLongWord   = LongWord(High(LongWord));

```

```

18774: MinLongInt    = LongInt(Low(LongInt));
18775: MaxLongInt    = LongInt(High(LongInt));
18776: MinInt64      = Int64(Low(Int64));
18777: MaxInt64      = Int64(High(Int64));
18778: MinInteger    = Integer(Low(Integer));
18779: MaxInteger    = Integer(High(Integer));
18780: MinCardinal   = Cardinal(Low(Cardinal));
18781: MaxCardinal   = Cardinal(High(Cardinal));
18782: MinNativeUInt = NativeUInt(Low(NativeUInt));
18783: MaxNativeUInt = NativeUInt(High(NativeUInt));
18784: MinNativeInt  = NativeInt(Low(NativeInt));
18785: MaxNativeInt  = NativeInt(High(NativeInt));
18786: Function CosH( const Z : Float) : Float;
18787: Function SinH( const Z : Float) : Float;
18788: Function TanH( const Z : Float) : Float;
18789:
18790:
18791: //*****from DMath.Dll Lib of types.inc in source\dmath_dll
18792: InvLn2          = 1.44269504088896340736; { 1/Ln(2) }
18793: InvLn10         = 0.43429448190325182765; { 1/Ln(10) }
18794: TwoPi          = 6.28318530717958647693; { 2*Pi }
18795: PiDiv2         = 1.57079632679489661923; { Pi/2 }
18796: SqrtPi         = 1.77245385090551602730; { Sqrt(Pi) }
18797: Sqrt2Pi        = 2.50662827463100050242; { Sqrt(2*Pi) }
18798: InvSqrt2Pi     = 0.39894228040143267794; { 1/Sqrt(2*Pi) }
18799: LnSqrt2Pi     = 0.91893853320467274178; { Ln(Sqrt(2*Pi)) }
18800: Ln2PiDiv2     = 0.91893853320467274178; { Ln(2*Pi)/2 }
18801: Sqrt2          = 1.41421356237309504880; { Sqrt(2) }
18802: Sqrt2Div2     = 0.70710678118654752440; { Sqrt(2)/2 }
18803: Gold           = 1.61803398874989484821; { Golden Mean = (1 + Sqrt(5))/2 }
18804: CGold          = 0.38196601125010515179; { 2 - GOLD }
18805: MachEp         = 2.220446049250313E-16; { 2^(-52) }
18806: NumMax         = 1.797693134862315E+308; { 2^1024 }
18807: MinNum         = 2.225073858507202E-308; { 2^(-1022) }
18808: MaxLog          = 709.7827128933840;
18809: MinLog          = -708.3964185322641;
18810: MaxFac          = 170;
18811: MaxGam          = 171.624376956302;
18812: MaxLgm          = 2.556348E+305;
18813: SingleCompareDelta = 1.0E-34;
18814: DoubleCompareDelta = 1.0E-280;
18815: {$IFDEF CLR}
18816: ExtendedCompareDelta = DoubleCompareDelta;
18817: {$ELSE}
18818: ExtendedCompareDelta = 1.0E-4400;
18819: {$ENDIF}
18820: Bytes1KB         = 1024;
18821: Bytes1MB         = 1024 * Bytes1KB;
18822: Bytes1GB         = 1024 * Bytes1MB;
18823: Bytes64KB        = 64 * Bytes1KB;
18824: Bytes64MB        = 64 * Bytes1MB;
18825: Bytes2GB         = 2 * LongWord(Bytes1GB);
18826: clBlack32'        = $FF000000 );
18827: clDimGray32'      = $FF3F3F3F );
18828: clGray32'         = $FF7F7F7F );
18829: clLightGray32'    = $FFBFBFBF );
18830: clWhite32'        = $FFFFFF );
18831: clMaroon32'       = $FF7F0000 );
18832: clGreen32'        = $FF007F00 );
18833: clOlive32'        = $FF7F7F00 );
18834: clNavy32'         = $FF00007F );
18835: clPurple32'       = $FF7F007F );
18836: clTeal32'         = $FF007F7F );
18837: clRed32'          = $FFFF0000 );
18838: clLime32'         = $FF00FF00 );
18839: clYellow32'       = $FFFFFF00 );
18840: clBlue32'         = $FF0000FF );
18841: clFuchsia32'      = $FFFF00FF );
18842: clAqua32'         = $FF00FFFF );
18843: clAliceBlue32'    = $FFF0F8FF );
18844: clAntiqueWhite32' = $FFFAEBD7 );
18845: clAquamarine32'   = $FF7FFF04 );
18846: clAzure32'        = $FFF0FFFF );
18847: clBeige32'        = $FFF5F5DC );
18848: clBisque32'       = $FFFFE4C4 );
18849: clBlancheDalmond32' = $FFFEBECD );
18850: clBlueViolet32'   = $FF8A2BE2 );
18851: clBrown32'        = $FFA52A2A );
18852: clBurlyWood32'    = $FFDEB887 );
18853: clCadetblue32'    = $FF5F9EA0 );
18854: clChartreuse32'   = $FF7FFF00 );
18855: clChocolate32'   = $FFD2691E );
18856: clCoral32'        = $FFFF7F50 );
18857: clCornflowerBlue32' = $FF6495ED );
18858: clCornSilk32'      = $FFFFF8DC );
18859: clCrimson32'      = $FFDC143C );
18860: clDarkBlue32'      = $FF00008B );
18861: clDarkCyan32'      = $FF008B8B );
18862: clDarkGoldenRod32' = $FFB8860B );

```

```
18863:    clDarkGray32', $FFA9A9A9 );
18864:    clDarkGreen32', $FF006400 );
18865:    clDarkGrey32', $FFA9A9A9 );
18866:    clDarkKhaki32', $FFBD876B );
18867:    clDarkMagenta32', $FF8B008B );
18868:    clDarkOliveGreen32', $FF556B2F );
18869:    clDarkOrange32', $FFFF8C00 );
18870:    clDarkOrchid32', $FF9932CC );
18871:    clDarkRed32', $FF8B0000 );
18872:    clDarkSalmon32', $FFE9967A );
18873:    clDarkSeaGreen32', $FF8FBBC8F );
18874:    clDarkSlateBlue32', $FF483D8B );
18875:    clDarkSlateGray32', $FF2F4F4F );
18876:    clDarkSlateGrey32', $FF2F4F4F );
18877:    clDarkTurquoise32', $FF00CED1 );
18878:    clDarkViolet32', $FF9400D3 );
18879:    clDeepPink32', $FFFF1493 );
18880:    clDeepSkyBlue32', $FF00BFFF );
18881:    clDodgerBlue32', $FF1E90FF );
18882:    clFireBrick32', $FFB22222 );
18883:    clFloralWhite32', $FFFFFAF0 );
18884:    clGainsboro32', $FFDCDCDC );
18885:    clGhostWhite32', $FFF8F8FF );
18886:    clGold32', $FFFFD700 );
18887:    clGoldenRod32', $FFDAE520 );
18888:    clGreenYellow32', $FFADFF2F );
18889:    clGrey32', $FF808080 );
18890:    clHoneyDew32', $FFF0FFF0 );
18891:    clHotPink32', $FFFF69B4 );
18892:    clIndianRed32', $FFCD5C5C );
18893:    clIndigo32', $FF4B0082 );
18894:    clIvory32', $FFFFFFF0 );
18895:    clKhaki32', $FFF0E68C );
18896:    clLavender32', $FFE6E6FA );
18897:    clLavenderBlush32', $FFFFFF0F5 );
18898:    clLawnGreen32', $FF7CFC00 );
18899:    clLemonChiffon32', $FFFFFFFACD );
18900:    clLightBlue32', $FFADD8E6 );
18901:    clLightCoral32', $FFF08080 );
18902:    clLightCyan32', $FFE0FFF );
18903:    clLightGoldenRodYellow32', $FFFAFAD2 );
18904:    clLightGreen32', $FF90EE90 );
18905:    clLightGrey32', $FFD3D3D3 );
18906:    clLightPink32', $FFFFB6C1 );
18907:    clLightSalmon32', $FFFA07A );
18908:    clLightSeagreen32', $FF20B2AA );
18909:    clLightSkyblue32', $FF87CEFA );
18910:    clLightSlategray32', $FF778899 );
18911:    clLightSlategrey32', $FF778899 );
18912:    clLightSteelblue32', $FFB0C4DE );
18913:    clLightYellow32', $FFFFFFE0 );
18914:    clLtGray32', $FFC0C0C0 );
18915:    clMedGray32', $FFA0A0A4 );
18916:    clDkGray32', $FF808080 );
18917:    clMoneyGreen32', $FFC0DCC0 );
18918:    clLegacySkyBlue32', $FFA6CAF0 );
18919:    clCream32', $FFFFFFBF0 );
18920:    clLimeGreen32', $FF32CD32 );
18921:    clLinen32', $FFFAF0E6 );
18922:    clMediumAquamarine32', $FF66CDAA );
18923:    clMediumBlue32', $FF0000CD );
18924:    clMediumOrchid32', $FFBA55D3 );
18925:    clMediumPurple32', $FF9370DB );
18926:    clMediumSeaGreen32', $FF3CB371 );
18927:    clMediumSlateBlue32', $FF7B68EE );
18928:    clMediumSpringGreen32', $FF00FA9A );
18929:    clMediumTurquoise32', $FF48D1CC );
18930:    clMediumVioletRed32', $FFC71585 );
18931:    clMidnightBlue32', $FF191970 );
18932:    clMintCream32', $FFFFFFFFFF );
18933:    clMistyRose32', $FFFE4E1 );
18934:    clMoccasin32', $FFFFFF4B5 );
18935:    clNavajoWhite32', $FFFFDEAD );
18936:    clOldLace32', $FFFDF5E6 );
18937:    clOliveDrab32', $FF6B8E23 );
18938:    clOrange32', $FFFA500 );
18939:    clOrangeRed32', $FFFF4500 );
18940:    clOrchid32', $FFDA70D6 );
18941:    clPaleGoldenRod32', $FFEEE8AA );
18942:    clPaleGreen32', $FF98FB98 );
18943:    clPaleTurquoise32', $FFAFEEEE );
18944:    clPaleVioletred32', $FFDB7093 );
18945:    clPapayaWhip32', $FFFFFFFD5 );
18946:    clPeachPuff32', $FFFFDAB9 );
18947:    clPeru32', $FFCD853F );
18948:    clPlum32', $FFDDA0DD );
18949:    clPowderBlue32', $FFB0E0E6 );
18950:    clRosyBrown32', $FFBC8F8F );
18951:    clRoyalBlue32', $FF4169E1 );
```

```

18952:   clSaddleBrown32', $FF8B4513 ) );
18953:   clSalmon32', $FFFA8072 );
18954:   clSandyBrown32', $FFF4A460 );
18955:   clSeaGreen32', $FF2E8B57 );
18956:   clSeaShell132', $FFFFF5EE );
18957:   clSienna32', $FFA0522D );
18958:   clSilver32', $FFC0C0C0 );
18959:   clSkyblue32', $FF87CEEB );
18960:   clSlateBlue32', $FF6A5ACD );
18961:   clSlateGray32', $FF708090 );
18962:   clSlateGrey32', $FF708090 );
18963:   clSnow32', $FFFFFFFAFA );
18964:   clSpringgreen32', $FF00FF7F );
18965:   clSteelblue32', $FF4682B4 );
18966:   clTan32', $FFD2B48C );
18967:   clThistle32', $FFD8BD8 );
18968:   clTomato32', $FFFF6347 );
18969:   clTurquoise32', $FF40E0D0 );
18970:   clViolet32', $FFEE82EE );
18971:   clWheat32', $FFF5DEB3 );
18972:   clWhitesmoke32', $FFF5F5F5 );
18973:   clYellowgreen32', $FF9ACD32 );
18974:   clTrWhite32', $7FFFFFFF );
18975:   clTrBlack32', $7F000000 );
18976:   clTrRed32', $7FFF0000 );
18977:   clTrGreen32', $7F00FF00 );
18978:   clTrBlue32', $7F0000FF );
18979: // Fixed point math constants
18980: FixedOne = $10000; FixedHalf = $7FFF;
18981: FixedPI = Round(PI * FixedOne);
18982: FixedToFloat = 1/FixedOne;
18983:
18984: Special Types
18985: ****
18986: type Complex = record
18987:   X, Y : Float;
18988: end;
18989: type TVector      = array of Float;
18990: TIntVector     = array of Integer;
18991: TCompVector    = array of Complex;
18992: TBoolVector    = array of Boolean;
18993: TStrVector     = array of String;
18994: TMatrix         = array of TVector;
18995: TIntMatrix     = array of TIntVector;
18996: TCompMatrix    = array of TCompVector;
18997: TBoolMatrix    = array of TBoolVector;
18998: TStrMatrix     = array of TStrVector;
18999: TByteArray     = array[0..32767] of byte; !
19000: THexArray      = array [0..15] of Char; // = '0123456789ABCDEF';
19001: TBitmapStyle   = (bsNormal, bsCentered, bsStretched);
19002: T2StringArray   = array of array of string;
19003: T2IntegerArray = array of array of integer;
19004: AddTypeS('INT_PTR', 'Integer
19005: AddTypeS('LONG_PTR', 'Integer
19006: AddTypeS('UINT_PTR', 'Cardinal
19007: AddTypeS('ULONG_PTR', 'Cardinal
19008: AddTypeS('DWORD_PTR', 'ULONG_PTR
19009: TIntegerDynArray', 'array of Integer
19010: TCardinalDynArray', 'array of Cardinal
19011: TWordDynArray', 'array of Word
19012: TSmallIntDynArray', 'array of SmallInt
19013: TByteDynArray', 'array of Byte
19014: TShortIntDynArray', 'array of ShortInt
19015: TInt64DynArray', 'array of Int64
19016: TLongWordDynArray', 'array of LongWord
19017: TSsingleDynArray', 'array of Single
19018: TDoubleDynArray', 'array of Double
19019: TBooleanDynArray', 'array of Boolean
19020: TStringDynArray', 'array of string
19021: TWideStringDynArray', 'array of WideString
19022: TDynByteArray   = array of Byte;
19023: TDynShortintArray= array of Shortint;
19024: TDynSmallintArray= array of Smallint;
19025: TDynWordArray   = array of Word;
19026: TDynIntegerArray= array of Integer;
19027: TDynLongintArray= array of Longint;
19028: TDynCardinalArray= array of Cardinal;
19029: TDynInt64Array   = array of Int64;
19030: TDynExtendedArray= array of Extended;
19031: TDynDoubleArray  = array of Double;
19032: TDynSingleArray  = array of Single;
19033: TDynFloatArray   = array of Float;
19034: TDynPointerArray = array of Pointer;
19035: TDynStringArray  = array of string;
19036: TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
19037:   ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
19038: TSynSearchOptions = set of TSynSearchOption;
19039:
19040:
```

```

19041: /* Project : Base Include RunTime Lib for maxBox *Name: pas_includebox.inc
19042: -----
19043: procedure drawPolygon(vPoints: TXYVector; cFrm: TForm);
19044: procedure drawPlot(vPoints: TXYVector; cFrm: TForm; vcolor: integer);
19045: procedure SaveCanvas(vCanvas: TCanvas; FileName: string);
19046: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
19047: function CheckStringSum(vstring: string): integer;
19048: function HexToInt(HexNum: string): LongInt;
19049: function IntToBin(Int: Integer): String;
19050: function BinToInt(Binary: String): Integer;
19051: function HexToBin(HexNum: string): string; external2;
19052: function BinToHex(Binary: String): string;
19053: function IntToFloat(i: Integer): double;
19054: function AddThousandSeparator(S: string; myChr: Char): string;
19055: function Max3(const X,Y,Z: Integer): Integer;
19056: procedure Swap(var X,Y: char); // faster without inline
19057: procedure ReverseString(var S: String);
19058: function CharToHexStr(Value: Char): string;
19059: function CharToUniCode(Value: Char): string;
19060: function Hex2Dec(Value: Str002): Byte;
19061: function HexStrCodeToStr(Value: string): string;
19062: function HexToStr(i: integer; value: string): string;
19063: function UniCodeToStr(Value: string): string;
19064: function CRC16(statement: string): string;
19065: function SearchForSubstrings(aStrList: TStrings; aSearchStr1, aSearchStr2: string): string;
19066: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string);
19067: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
19068: Procedure ExecuteCommand(executeFile, paramString: string);
19069: Procedure ShellExecuteAndWait(executeFile, paramString: string);
19070: function ExecConsoleApp(const AppName, Parameters: String; AppOutput: TStrings): DWORD;
19071: procedure SearchAndOpenDoc(vfilenamepath: string);
19072: procedure ShowInterfaces(myFile: string);
19073: function Fact2(av: integer): extended;
19074: Function BoolToStr(B: Boolean): string;
19075: Function GCD(x, y: LongInt) : LongInt;
19076: function LCM(m,n: longint): longint;
19077: function GetASCII: string;
19078: function GetItemHeight(Font: TFont): Integer;
19079: function myPlaySound(s: pchar; flag,syncflag: integer): boolean;
19080: function myGetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
19081: function getHINSTANCE: longword;
19082: function getHMODULE: longword;
19083: function GetASCII: string;
19084: function ByteIsOk(const AByte: string; var VB: Byte): boolean;
19085: function WordIsOk(const AWord: string; var VW: Word): boolean;
19086: function TwentyFourBitValueIsOk(const AValue: string; var VI: Integer): boolean;
19087: function LongIsOk(const ALong: string; var VC: Cardinal): boolean;
19088: function SafeStr(const s: string): string;
19089: function ExtractUrlPath(const FileName: string): string;
19090: function ExtractUrlName(const FileName: string): string;
19091: function IsInternet: boolean;
19092: function RotateLeft1Bit_u32( Value: uint32): uint32;
19093: procedure LinearRegression(const KnownY:array of Double;const KnownX: array of Double;NData:Int;var LF:TStLinEst; ErrorStats : Boolean);
19094: procedure getEnvironmentInfo;
19095: procedure AntiFreeze;
19096: function GetCPUSpeed: Double;
19097: function IsVirtualPcGuest : Boolean;
19098: function IsVmWareGuest : Boolean;
19099: procedure StartSerialDialog;
19100: function IsWoW64: boolean;
19101: function IsWow64String(var s: string): Boolean;
19102: procedure StartThreadDemo;
19103: Function RGB(R,G,B: Byte): TColor;
19104: Function Sendln(amess: string): boolean;
19105: Procedure maxBox;
19106: Function AspectRatio(aWidth, aHeight: Integer): String;
19107: function wget(aURL, afile: string): boolean;
19108: procedure PrintList(Value: TStringList);
19109: procedure PrintImage(aValue: TBitmap; Style: TBitmapStyle);
19110: procedure getEnvironmentInfo;
19111: procedure AntiFreeze;
19112: function getBitmap(apath: string): TBitmap;
19113: procedure ShowMessageBig(const aText : string);
19114: function YesNoDialog(const ACaption, AMsg: string): boolean;
19115: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);
19116: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
19117: //function myStrToBytes(const Value: String): TBytes;
19118: //function myBytesToStr(const Value: TBytes): String;
19119: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
19120: function getBitmap(apath: string): TBitmap;
19121: procedure ShowMessageBig(const aText : string);
19122: Function StrToBytes(const Value: String): TBytes;
19123: Function BytesToStr(const Value: TBytes): String;
19124: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
19125: function ReverseDNSLookup(const IPAdrs:String;const DNSServer:String;Timeout,Retries:Int;var HostName:String):Bool;
19126: function FindInPaths(const fileName, paths : String) : String;
19127: procedure initHexArray(var hexn: THexArray);

```

```

19128: function josephusG(n,k: integer; var graphout: string): integer;
19129: function isPowerof2(num: int64): boolean;
19130: function powerOf2(exponent: integer): int64;
19131: function getBigPI: string;
19132: procedure MakeSound(Frequency{Hz}, Duration{mSec}: Integer; Volume: TVolumeLevel; savefilePath: string);
19133: function GetASCIILine: string;
19134: procedure MakeComplexSound(N:integer{stream # to use}; freqlist:TStrings; Duration{mSec}: Integer;
19135:                                pinknoise: boolean; shape: integer; Volume: TVolumeLevel);
19136: procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
19137: procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
19138: function mapfunc(ax, in_min, in_max, out_min, out_max: integer): integer;
19139: function mapmax(ax, in_min, in_max, out_min, out_max: integer): integer;
19140: function isKeyPressed: boolean;
19141: function Keypress: boolean;
19142: procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
19143: function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
19144: function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
19145: function GetOSName: string;
19146: function GetOSVersion: string;
19147: function GetOSNumber: string;
19148: function GetEnvironmentString: string;
19149: procedure StrReplace(var Str: String; Old, New: String);
19150: procedure SendEmail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
19151: function GetTeamViewerID: string;
19152: Procedure RecurseDirectory(Dir:String; IncludeSubs:boolean; callback:TFileCallbackProcedure);
19153: Procedure RecurseDirectory2(Dir : String; IncludeSubs : boolean);
19154: procedure WinInet_HttpGet(const Url: string; Stream:TStream);
19155: procedure GetQrCode2(Width,Height: Word;Correct_Level:string;const Data:string;apath:string);
19156: function StartSocketService: Boolean;
19157: procedure StartSocketServiceForm;
19158: function GetFileList(FileList: TStringlist; apath: string): TStringlist;
19159: function GetFileList1(apath: string): TStringlist;
19160: procedure LetfileList(FileList: TStringlist; apath: string);
19161: procedure StartWeb(aurl: string);
19162: function GetTodayFiles(startdir, anask: string): TStringlist;
19163: function PortTCPISOpen(dwPort: Word; ipAddressStr: String): boolean;
19164: function JavahashCode(val: string): Integer;
19165: procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
19166: procedure SaveFileToFile2(const Data: Sysutils.TBytes; const FileName: string);
19167: Procedure HideWindowForSeconds(secs: integer); //3 seconds
19168: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm); //3 seconds
19169: Procedure ConvertToGray(Cnv: TCanvas);
19170: function GetFileDate(aFile:string; aWithTime:Boolean):string;
19171: procedure ShowMemory;
19172: function ShowMemory2: string;
19173: function getHostIP: string;
19174: procedure ShowBitmap(bmap: TBitmap);
19175: function GetOsVersionInfo: TOSVersionInfo; //thx to wischnewski
19176: function CreateDBGridForm(dblist: TStringList): TListBox;
19177: function isService: boolean;
19178: function isApplication: boolean;
19179: function isTerminalSession: boolean;
19180:
19181:
19182: // News of 3.9.8 up
19183: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
19184: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
19185: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
19186: JvChart - TJvChart Component - 2009 Public
19187: MemoryLeakReport in ini-file (MEMORYREPORT=Y)
19188: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
19189: TAdoQuery.SQL.Add() fixed, ShlwAPI extensions, Indy HTTPHeader Extensions
19190: DMATH DLL included incl. Demos
19191: Interface Navigator menu/View/Intf Navigator
19192: Unit Explorer menu/Debug/Units Explorer
19193: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maxcel
19194: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
19195: Script History to 9 Files WebServer light /Options/Addons/WebServer
19196: Full Text Finder, JVSimLogic Simulator Package
19197: Halt-Stop Program in Menu, WebServer2, Stop Event ,
19198: Conversion Routines, Prebuild Forms, CodeSearch
19199: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
19200: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
19201: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
19202: JvChart - TJvChart Component - 2009 Public, mXGames, JvgXMLSerializer, TJvPaintFX
19203: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib
19204: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
19205: IDE Reflection API, Session Service Shell S3
19206: additional SynEdit API, isKeyPressed Routine, Bookmarks, OpenToolsAPI Catalog (OTAC)
19207: Class TMonitor, Configuration Tutorial maxbox_starter25.pdf, Chess.dll Game
19208: arduino map() function, PRMRandom Generator
19209: StBarCode Lib, StreamReaderClass, BarCode Package, Astro Package
19210: more ShellAPI, add 32 more units, Simulated Annealing, GenAlgo
19211: REST Test Lib, Multilang Component, Forth Interpreter
19212: New Macros, Sendmail (instant email), DevCUnits, Tetris Addon
19213: DCOM, MDAC, MIDI, TLS support, Posmarks, Utils Addon
19214: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
19215: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
19216: First LCL of Lazarus, CmdLine API, ToDo List, 36 more Units preCompiled

```

```

19217: QRCode Service, add more CFunctions like CDateTime of Synapse
19218: Gamma Functions, IndyPackage4, HotLog Threadable, FormTemplateLibrary FTL
19219: Nonlinear regression, ADO Workbench Addon, Assign fixing, IntfNavigator fixing, Applet
19220: 30 more Units preCompiled, QRCode Indy Service, more CFunctions like CFill or SRand
19221: RestartDialog, RTF, SQL Scanner, RichEdit, 15 more Units
19222: Tool Section, SOAP Tester, Hot Log Logger2, TCPPortScan, 28 more Units
19223: BOLD Package, Indy Package5, maTRIX. MATHEMAX
19224: SPS Utils WDOS, Plc BitBus (PetriNet), 40 more units
19225: emax layers: system-package-component-unit-class-function-block
19226: HighPrecision Timers, Indy Package6, AutoDetect, UltraForms
19227: Reversi, GOL, bugfixing, 8 more units, Tutorial 24 Clean Code
19228: Tutorial 18_3 RGB LED, OpenGL Geometry, maxpix, statictext
19229: OpenGL Game Demo: ..Options/Add Ons/Reversi
19230: IBUtils Refactor, InterBase Package, DotNet Routines (JvExControls)
19231: add 31 units, mX4 Introduction Paper, more Socket&Streams, ShortString Routines
19232: 7% performance gain (hot spot profiling)
19233: PEP -Pascal Education Program, GSM Module, CGI, PHP Runner
19234: add 42 + 22 (64 units), memcached database, autobookmark, Alcinoe PAC, IPC Lib
19235: Orpheus PAC, AsyncFree Library advapi32 samples, FirebirdExp+MySQL units
19236: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
19237:
19238: add routines in 3.9.7.5
19239: 097: procedure RIRegister_BarCodeScanner_Routines(S: TPSEexec);
19240: 996: procedure RIRegister_DBCtrls_Routines(S: TPSEexec);
19241: 069: procedure RIRegister_IdStrings_Routines(S: TPSEexec);
19242: 516: procedure RIRegister_JclMultimedia_Routines(S: TPSEexec);
19243: 215: procedure RIRegister_PNGLoader_Routines(S: TPSEexec);
19244: 374: procedure RIRegister_SerDlgls_Routines(S: TPSEexec);
19245: 777: procedure RIRegister_LinarBitmap_Routines(S: TPSEexec);
19246:
19247: ////////////////////////////////////////////////////////////////// TestUnits //////////////////////////////////////////////////////////////////
19248: SelftestPEM;
19249: SelfTestCFundamentUtils;
19250: SelfTestCFileUtils;
19251: SelfTestCDatetime;
19252: SelfTestCTimer;
19253: SelfTestCRandom;
19254: Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter'
19255:           Assert(WinPathToUnixPath('\c\d\f') = '/c/d/f', 'WinPathToUnixPath'
19256:
19257: // Note: There's no need for installing a client certificate in the
19258: // webbrowser. The server asks the webbrowser to send a certificate but
19259: // if nothing is installed the software will work because the server
19260: // doesn't check to see if a client certificate was supplied. If you want you can install:
19261: // file: c_cacert.p12
19262: // password: c_cakey
19263:
19264: TGraphicControl = class(TControl)
19265: private
19266:   FCanvas: TCanvas;
19267:   procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
19268: protected
19269:   procedure Paint; virtual;
19270:   property Canvas: TCanvas read FCanvas;
19271: public
19272:   constructor Create(AOwner: TComponent); override;
19273:   destructor Destroy; override;
19274: end;
19275:
19276: TCustomControl = class(TWinControl)
19277: private
19278:   FCanvas: TCanvas;
19279:   procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
19280: protected
19281:   procedure Paint; virtual;
19282:   procedure PaintWindow(DC: HDC); override;
19283:   property Canvas: TCanvas read FCanvas;
19284: public
19285:   constructor Create(AOwner: TComponent); override;
19286:   destructor Destroy; override;
19287: end;
19288: RegisterPublishedProperties;
19289: ('ONCHANGE', 'TNotifyEvent', iptrw);
19290: ('ONCLICK', 'TNotifyEvent', iptrw);
19291: ('ONDBLCLICK', 'TNotifyEvent', iptrw);
19292: ('ONENTER', 'TNotifyEvent', iptrw);
19293: ('ONEXIT', 'TNotifyEvent', iptrw);
19294: ('ONKEYDOWN', 'TKeyEvent', iptrw);
19295: ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
19296: ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
19297: ('ONMOUSEMOVE', 'TMouseMoveEvent', iptrw);
19298: ('ONMOUSEUP', 'TMouseEvent', iptrw);
19299: //*****
19300: // To stop the while loop, click on Options>Show Include (boolean switch)!
19301: Control a loop in a script with a form event:
19302: IncludeON; //control the while loop
19303: while maxform1.ShowInclude1.checked do begin //menu event Options>Show Include
19304:
19305: //-----

```

```

19306: //*****mX4 ini-file Configuration*****
19307: -----
19308:   using config file maxboxdef.ini      menu/Help/Config File
19309:
19310: //*** Definitions for maxBox mX3 ***
19311: [FORM]
19312: LAST_FILE=E:\maxbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
19313: FONTSIZE=14
19314: EXTENSION=txt
19315: SCREENX=1386
19316: SCREENY=1077
19317: MEMHEIGHT=350
19318: PRINTFONT=Courier New //GUI Settings
19319: LINENUMBERS=Y //line numbers at gutter in editor at left side
19320: EXCEPTIONLOG=Y //store excepts and success in 2 log files see below! -menu Debug>Show Last Exceptions
19321: EXECUTESHELL=Y //prevents execution of ExecuteShell() or ExecuteCommand()
19322: BOOTSCRIPT=Y //enabling load a boot script
19323: MEMORYREPORT=Y //shows memory report on closing maxBox
19324: MACRO=Y //expand macros (see below) in code e.g. #path:E:\maxbox3\mXGit39988\maxbox3\docs\
19325: NAVIGATOR=N //shows function list at the right side of editor
19326: NAVWIDTH=350 //width of the right side interface list <CTRL L> >=200
19327: AUTOBOOKMARK=Y //sets on all functions a bookmark to jump
19328: [WEB]
19329: IPORT=8080 //for internal webserver - menu /Options/Add Ons/WebServer
19330: IPHOST=192.168.1.53
19331: ROOTCERT=filepathY
19332: SCERT=filepathY
19333: RSAKEY=filepathY
19334: VERSIONCHECK=Y
19335:
19336: using Logfile: maxboxlog.log , Exceptionlogfile: maxboxerrorlog.txt
19337:
19338: Also possible to set report memory in script to override ini setting
19339: procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
19340:
19341:
19342: After Change the ini file you can reload the file with ./Help/Config Update
19343:
19344: -----
19345: //*****mX4 maildef.ini ini-file Configuration*****
19346: -----
19347: //*** Definitions for maxMail ***
19348: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
19349: [MAXMAIL]
19350: HOST=getmail.softwareschule.ch
19351: USER=mailusername
19352: PASS=password
19353: PORT=110
19354: SSL=Y
19355: BODY=Y
19356: LAST=5
19357:
19358: ADO Connection String:
19359: Provider=MSDASQL;DSN=mx3base;Uid=sa;Pwd=admin
19360:
19361: \452_dbtreeview2access.txt
19362: program TestDbTreeViewMainForm2_ACCESS;
19363:   ConnectionString:='Provider=Microsoft.Jet.OLEDB.4.0;Data Source='
19364:           +Exepath+'\examples\detail.mdb;Persist Security Info=False';
19365:
19366: OpenSSL Lib: unit ssl_openssl_lib;
19367: {$IFDEF CIL}
19368: const
19369: {$IFDEF LINUX}
19370: DLLSSLName = 'libssl.so';
19371: DLLUtilName = 'libcrypto.so';
19372: {$ELSE}
19373: DLLSSLName = 'ssleay32.dll';
19374: DLLUtilName = 'libeay32.dll';
19375: {$ENDIF}
19376: {$ELSE}
19377: var
19378: {$IFDEF MSWINDOWS}
19379: {$IFDEF DARWIN}
19380: DLLSSLName: string = 'libssl.dylib';
19381: DLLUtilName: string = 'libcrypto.dylib';
19382: {$ELSE}
19383: DLLSSLName: string = 'libssl.so';
19384: DLLUtilName: string = 'libcrypto.so';
19385: {$ENDIF}
19386: {$ELSE}
19387: DLLSSLName: string = 'ssleay32.dll';
19388: DLLSSLName2: string = 'libssl32.dll';
19389: DLLUtilName: string = 'libeay32.dll';
19390: {$ENDIF}
19391: {$ENDIF}
19392:
19393:
19394: -----

```

```

19395: //*****mX4 Macro Tags ****
19396: //-----
19397:
19398:   asm #name #hostmAPSN2APSN211le, #head,max: APSN21: 04.01.2014 19:05:50
19399:   E:\maxbox\maxbox3\docs\maxbox_extract_funcList399.txt end
19400: //Tag Macros
19401:
19402:   asm #name, #date, #host, #path, #file, #head, #sign #tech #net end
19403:
19404: //Tag Macros
19405: 10188: SearchAndCopy(memo1.lines, '#name', getUserNameWin, 11);
19406: 10189: SearchAndCopy(memo1.lines, '#date', datetimetoStr(now), 11);
19407: 10190: SearchAndCopy(memo1.lines, '#host', getComputerNameWin, 11);
19408: 10191: SearchAndCopy(memo1.lines, '#path', fpath, 11);
19409: 10192: SearchAndCopy(memo1.lines, '#file', fname, 11);
19410: 10199: SearchAndCopy(memo1.lines, '#fils', fname + '+SHA1(Act_Filename), 11);
19411: 10193: SearchAndCopy(memo1.lines, '#locs', intToStr(getCodeEnd), 11);
19412: 10194: SearchAndCopy(memo1.lines, '#perf', perfTime, 11);
19413: 10195: SearchAndCopy(memo1.lines, '#sign', Format('%s: %s', [getUserNameWin, getComputerNameWin, datetimetoStr(now)]);
19414: 10196: SearchAndCopy(memo1.lines, '#head', Format('%s: %s: %s %s ', [getUserNameWin, getComputerNameWin, datetimetoStr(now), Act_Filename]), 11);
19416: 10197: [getUserNameWin, getComputerNameWin, datetimetoStr(now), Act_Filename]], 11);
19417: 10198: SearchAndCopy(memo1.lines, '#tech', Format('perf: %s threads: %d %s %s', [perfTime, numProcessesThreads, getIPAddress(getComputerNameWin), timetoStr(time), mbVersion]]), 11);
19419: 10298: SearchAndCopy(memo1.lines, '#net', Format('DNS: %s; local IPs: %s; local IP: %s', [getDNS, GetLocalIPs, getIPAddress(getComputerNameWin)])), 10);
19421:
19422:
19423: //#tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
19424:
19425: //Replace Macros
19426:   SearchAndCopy(memo1.lines, '<TIME>', timetoStr(time), 6);
19427:   SearchAndCopy(memo1.lines, '<DATE>', datetoStr(date), 6);
19428:   SearchAndCopy(memo1.lines, '<PATH>', fpath, 6);
19429:   SearchAndCopy(memo1.lines, '<EXEPAUTH>', EXEPath, 9);
19430:   SearchAndCopy(memo1.lines, '<FILE>', fname, 6);
19431:   SearchAndCopy(memo1.lines, '<SOURCE>', ExePath + 'Source', 8);
19432:
19433:   SearchAndCopy(memo1.lines, '#tech!perf: threads: 2 192.168.1.53 19:05:50 3.9.9.84
19434:   [perfTime, numProcessesThreads, getIPAddress(getComputerNameWin), timetoStr(time), mbVersion]], 11);
19435: //#tech!84perf: threads: 2 192.168.1.53 19:05:50 3.9.9.84
19436:   SearchAndCopy(memo1.lines, 'maxbox_extract_funcList399.txt
19437:
19438: //-----
19439: //*****mX4 ToDo List Tags ..../HelpToDo List*****
19440: //-----
19441:
19442:   while I < sl.Count do begin
19443:     if MatchesMask(sl[I], '/? TODO ([a-z0-9_]*#[1-9]#)*:*') then
19444:       if MatchesMask(sl[I], '/? TODO (?*#?#)*:*') then
19445:         BreakupToDo(Filename, sl, I, 'TODO', True, True) // full info TODO
19446:       else if MatchesMask(sl[I], '/? DONE (?*#?#)*:*') then
19447:         BreakupToDo(Filename, sl, I, 'DONE', True, True) // full info DONE
19448:       else if MatchesMask(sl[I], '/? TODO (#?#)*:*') then
19449:         BreakupToDo(Filename, sl, I, 'TODO', False, True) // only priority info TODO
19450:       else if MatchesMask(sl[I], '/? DONE (#?#)*:*') then
19451:         BreakupToDo(Filename, sl, I, 'DONE', False, True) // only priority info DONE
19452:       else if MatchesMask(sl[I], '/?/*TODO*:*)' then
19453:         BreakupToDo(Filename, sl, I, 'TODO', False, False) // custom TODO
19454:       else if MatchesMask(sl[I], '/?/*DONE*:*)' then
19455:         BreakupToDo(Filename, sl, I, 'DONE', False, False) // custom DONE
19456:       Inc(I);
19457:     end;
19458:
19459:
19460: //-----
19461: //*****mX4 Public Tools API ****
19462: //-----
19463:   file : unit uPSI_fMain.pas;                                {$SOTAP} Open Tools API Catalog
19464:   // Those functions concern the editor and preprocessor, all of the IDE
19465:   Example: Call it with maxform1.InfolClick(self)
19466:   Note: Call all Methods with maxForm1., e.g.:
19467:           maxForm1.ShellStyle1Click(self);
19468:
19469: procedure SIRegister_fMain(CL: TPSPascalCompiler);
19470: begin
19471:   Const ('BYTECODE', 'String 'bytecode.txt'
19472:   Const ('PSTEXT', 'String PS Scriptfiles (*.txt)|*.TXT')
19473:   Const ('PSMODEL', 'String PS Modelfiles (*.uc)|*.UC')
19474:   Const ('PSPASCAL', 'String PS Pascalfiles (*.pas)|*.PAS')
19475:   Const ('PSINC', 'String PS Includes (*.inc)|*.INC')
19476:   Const ('DEFFILENAME', 'String 'firstdemo.txt'
19477:   Const ('DEFINIFILE', 'String 'maxboxdef.ini'
19478:   Const ('EXCEPTLOGFILE', 'String 'maxboxerrorlog.txt'
19479:   Const ('ALLFUNCTIONSLIST', 'String 'upsi_allfunctionslist.txt'
19480:   Const ('ALLFUNCTIONSLISTPDF', 'String 'maxbox_functions_all.pdf'
19481:   Const ('ALLOBJECTSLIST', 'String 'docs\VCL.pdf'
19482:   Const ('ALLRESOURCELIST', 'String 'docs\upsi_allresourceList.txt'

```

```
19483: Const('ALLUNITLIST','String 'docs\maxbox3_9.xml');
19484: Const('INCLUDEBOX','String 'pas_includebox.inc
19485: Const('BOOTSCRIPT','String 'maxbootscript.txt
19486: Const('MBVERSION','String '3.9.9.98
19487: Const('VERSION','String'3.9.9.98
19488: Const('MBVER','String '399
19489: Const('MBVERI','Integer'(399);
19490: Const('MBVERIALL','Integer'(39988);
19491: Const('EXENAME','String 'maxBox3.exe
19492: Const('MXSITE','String 'http://www.softwareschule.ch/maxbox.htm
19493: Const('MXVERSIONFILE','String 'http://www.softwareschule.ch/maxvfile.txt
19494: Const('MXVERSIONFILE2','String 'http://www.softwareschule.ch/maxvfile2.txt
19495: Const('MXINTERNETCHECK','String 'www.ask.com
19496: Const('MXMAIL','String 'max@kleiner.com
19497: Const('TAB','Char #'09);
19498: Const('CODECOMPLETION','String 'bds_delphi.dci
19499: SIRegister_TMaxForm1(CL);
19500: end;
19501:
19502: with FindClass('TForm','TMaxForm1') do begin
19503:   memo2', 'TMemo', iptrw);
19504:   memo1', 'TSynMemo', iptrw);
19505:   CB1SCList', 'TComboBox', iptrw);
19506:   mxNavigator', 'TComboBox', iptrw);
19507:   IPHost', 'string', iptrw);
19508:   IPPort', 'integer', iptrw);
19509:   COMPort', 'integer', iptrw); //3.9.6.4
19510:   Splitter1', 'TSplitter', iptrw);
19511:   PS3Script', 'TPSScript', iptrw);
19512:   PS3DllPlugin', 'TPSDllPlugin', iptrw);
19513:   MainMenul', 'TMainMenu', iptrw);
19514:   Programl', 'TMenuItem', iptrw);
19515:   Compilel', 'TMenuItem', iptrw);
19516:   Files1', 'TMenuItem', iptrw);
19517:   open1', 'TMenuItem', iptrw);
19518:   Save2', 'TMenuItem', iptrw);
19519:   Options1', 'TMenuItem', iptrw);
19520:   Savebefore1', 'TMenuItem', iptrw);
19521:   Largefont1', 'TMenuItem', iptrw);
19522:   sBytecode1', 'TMenuItem', iptrw);
19523:   Saveas3', 'TMenuItem', iptrw);
19524:   Clear1', 'TMenuItem', iptrw);
19525:   Slinenumbers1', 'TMenuItem', iptrw);
19526:   About1', 'TMenuItem', iptrw);
19527:   Search1', 'TMenuItem', iptrw);
19528:   SynPasSyn1', 'TSynPasSyn', iptrw);
19529:   memo1', 'TSynMemo', iptrw);
19530:   SynEditSearch1', 'TSynEditSearch', iptrw);
19531:   WordWrap1', 'TMenuItem', iptrw);
19532:   XManifest1', 'TXPManifest', iptrw);
19533:   SearchNext1', 'TMenuItem', iptrw);
19534:   Replace1', 'TMenuItem', iptrw);
19535:   PSImport_Controls1', 'TPSImport_Controls', iptrw);
19536:   PSImport_Classes1', 'TPSImport_Classes', iptrw);
19537:   ShowInclude1', 'TMenuItem', iptrw);
19538:   SynEditPrint1', 'TSynEditPrint', iptrw);
19539:   Printout1', 'TMenuItem', iptrw);
19540:   mnPrintColors1', 'TMenuItem', iptrw);
19541:   dlgFilePrint', 'TPrintDialog', iptrw);
19542:   dlgPrintFont1', 'TFontDialog', iptrw);
19543:   mnuPrintFont1', 'TMenuItem', iptrw);
19544:   Includel', 'TMenuItem', iptrw);
19545:   CodeCompletionList1', 'TMenuItem', iptrw);
19546:   IncludeList1', 'TMenuItem', iptrw);
19547:   ImageList1', 'TImageList', iptrw);
19548:   ImageList2', 'TImageList', iptrw);
19549:   CoolBar1', 'TCoolBar', iptrw);
19550:   ToolBar1', 'TToolBar', iptrw);
19551:   btnLoad', 'TToolButton', iptrw);
19552:   ToolButton2', 'TToolButton', iptrw);
19553:   btnFind', 'TToolButton', iptrw);
19554:   btnCompile', 'TToolButton', iptrw);
19555:   btnTrans', 'TToolButton', iptrw);
19556:   btnUseCase', 'TToolButton', iptrw); //3.8
19557:   toolbtnTutorial', 'TToolButton', iptrw);
19558:   btn6res', 'TToolButton', iptrw);
19559:   ToolButton5', 'TToolButton', iptrw);
19560:   ToolButton1', 'TToolButton', iptrw);
19561:   ToolButton3', 'TToolButton', iptrw);
19562:   statusBar1', 'TStatusBar', iptrw);
19563:   SaveOutput1', 'TMenuItem', iptrw);
19564:   ExportClipboard1', 'TMenuItem', iptrw);
19565:   Close1', 'TMenuItem', iptrw);
19566:   Manual1', 'TMenuItem', iptrw);
19567:   About2', 'TMenuItem', iptrw);
19568:   loadLastfile1', 'TMenuItem', iptrw);
19569:   imgLogo', 'TImage', iptrw);
19570:   cedebbug', 'TPSScriptDebugger', iptrw);
19571:   debugPopupMenu1', 'TPopupMenu', iptrw);
```

```
19572: BreakPointMenu', 'TMenuItem', iptrw);
19573: Decompile1', 'TMenuItem', iptrw);
19574: StepInto1', 'TMenuItem', iptrw);
19575: StepOut1', 'TMenuItem', iptrw);
19576: Reset1', 'TMenuItem', iptrw);
19577: DebugRun1', 'TMenuItem', iptrw);
19578: PSImport_ComObj1', 'TPSImport_ComObj', iptrw);
19579: PSImport_StdCtrls1', 'TPSImport_StdCtrls', iptrw);
19580: PSImport_Forms1', 'TPSImport_Forms', iptrw);
19581: PSImport_DateUtils1', 'TPSImport_DateUtils', iptrw);
19582: tutorial4', 'TMenuItem', iptrw);
19583: ExporttoClipboard1', 'TMenuItem', iptrw);
19584: ImportfromClipboard1', 'TMenuItem', iptrw);
19585: N4', 'TMenuItem', iptrw);
19586: N5', 'TMenuItem', iptrw);
19587: N6', 'TMenuItem', iptrw);
19588: ImportfromClipboard2', 'TMenuItem', iptrw);
19589: tutorial1', 'TMenuItem', iptrw);
19590: N7', 'TMenuItem', iptrw);
19591: ShowSpecChars1', 'TMenuItem', iptrw);
19592: OpenDirectory1', 'TMenuItem', iptrw);
19593: procMess', 'TMenuItem', iptrw);
19594: tbtnUseCase', 'TToolbutton', iptrw);
19595: ToolButton7', 'TToolbutton', iptrw);
19596: EditFont1', 'TMenuItem', iptrw);
19597: UseCase1', 'TMenuItem', iptrw);
19598: tutorial21', 'TMenuItem', iptrw);
19599: OpenUseCase1', 'TMenuItem', iptrw);
19600: PSImport_DB1', 'TPSImport_DB', iptrw);
19601: tutorial31', 'TMenuItem', iptrw);
19602: SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
19603: HTMLEntax1', 'TMenuItem', iptrw);
19604: ShowInterfaces1', 'TMenuItem', iptrw);
19605: Tutorial5', 'TMenuItem', iptrw);
19606: AllFunctionsList1', 'TMenuItem', iptrw);
19607: ShowLastException1', 'TMenuItem', iptrw);
19608: PlayMP31', 'TMenuItem', iptrw);
19609: SynTeXSyn1', 'TSynTeXSyn', iptrw);
19610: texSyntax1', 'TMenuItem', iptrw);
19611: N8', 'TMenuItem', iptrw);
19612: GetEMails1', 'TMenuItem', iptrw);
19613: SynCppSyn1', 'TSynCppSyn', iptrw);
19614: CSyntax1', 'TMenuItem', iptrw);
19615: Tutorial6', 'TMenuItem', iptrw);
19616: New1', 'TMenuItem', iptrw);
19617: AllObjectsList1', 'TMenuItem', iptrw);
19618: LoadBytecode1', 'TMenuItem', iptrw);
19619: CipherFile1', 'TMenuItem', iptrw);
19620: N9', 'TMenuItem', iptrw);
19621: N10', 'TMenuItem', iptrw);
19622: Tutorial11', 'TMenuItem', iptrw);
19623: Tutorial71', 'TMenuItem', iptrw);
19624: UpdateService1', 'TMenuItem', iptrw);
19625: PascalSchool1', 'TMenuItem', iptrw);
19626: Tutorial81', 'TMenuItem', iptrw);
19627: DelphiSite1', 'TMenuItem', iptrw);
19628: Output1', 'TMenuItem', iptrw);
19629: TerminalStyle1', 'TMenuItem', iptrw);
19630: ReadOnly1', 'TMenuItem', iptrw);
19631: Shellstyle1', 'TMenuItem', iptrw);
19632: BigScreen1', 'TMenuItem', iptrw);
19633: Tutorial91', 'TMenuItem', iptrw);
19634: SaveOutput2', 'TMenuItem', iptrw);
19635: N11', 'TMenuItem', iptrw);
19636: SaveScreenshot', 'TMenuItem', iptrw);
19637: Tutorial101', 'TMenuItem', iptrw);
19638: SQLSyntax1', 'TMenuItem', iptrw);
19639: SynSQLSyn1', 'TSynSQLSyn', iptrw);
19640: Console1', 'TMenuItem', iptrw);
19641: SynXMLSyn1', 'TSynXMLSyn', iptrw);
19642: XMLSyntax1', 'TMenuItem', iptrw);
19643: ComponentCount1', 'TMenuItem', iptrw);
19644: NewInstance1', 'TMenuItem', iptrw);
19645: toolbtnTutorial', 'TToolbutton', iptrw);
19646: Memory1', 'TMenuItem', iptrw);
19647: SynJavaSyn1', 'TSynJavaSyn', iptrw);
19648: JavaSyntax1', 'TMenuItem', iptrw);
19649: SyntaxCheck1', 'TMenuItem', iptrw);
19650: Tutorial10Statistics1', 'TMenuItem', iptrw);
19651: ScriptExplorer1', 'TMenuItem', iptrw);
19652: FormOutput1', 'TMenuItem', iptrw);
19653: ArduinoDump1', 'TMenuItem', iptrw);
19654: AndroidDump1', 'TMenuItem', iptrw);
19655: GotoEnd1', 'TMenuItem', iptrw);
19656: AllResourceList1', 'TMenuItem', iptrw);
19657: ToolButton4', 'TToolbutton', iptrw);
19658: tbtn6res', 'TToolbutton', iptrw);
19659: Tutorial11Forms1', 'TMenuItem', iptrw);
19660: Tutorial12SQL1', 'TMenuItem', iptrw);
```

```
19661: ResourceExplore1', 'TMenuItem', iptrw);
19662: Info1', 'TMenuItem', iptrw);
19663: N12', 'TMenuItem', iptrw);
19664: CryptoBox1', 'TMenuItem', iptrw);
19665: Tutorial13Ciphering1', 'TMenuItem', iptrw);
19666: CipherFile2', 'TMenuItem', iptrw);
19667: N13', 'TMenuItem', iptrw);
19668: ModulesCount1', 'TMenuItem', iptrw);
19669: AddOns2', 'TMenuItem', iptrw);
19670: N4GewinntGame1', 'TMenuItem', iptrw);
19671: DocuforAddOns1', 'TMenuItem', iptrw);
19672: Tutorial14Async1', 'TMenuItem', iptrw);
19673: Lessons15Review1', 'TMenuItem', iptrw);
19674: SynPHPSyn1', 'TSynPHPSyn', iptrw);
19675: PHPSyntax1', 'TMenuItem', iptrw);
19676: Breakpoint1', 'TMenuItem', iptrw);
19677: SerialRS2321', 'TMenuItem', iptrw);
19678: N14', 'TMenuItem', iptrw);
19679: SynCSSyn1', 'TSynCSSyn', iptrw);
19680: CSyntax2', 'TMenuItem', iptrw);
19681: Calculator1', 'TMenuItem', iptrw);
19682: tbtnSerial', 'TToolButton', iptrw);
19683: ToolButton8', 'TToolButton', iptrw);
19684: Tutorial151', 'TMenuItem', iptrw);
19685: N15', 'TMenuItem', iptrw);
19686: N16', 'TMenuItem', iptrw);
19687: ControlBar1', 'TControlBar', iptrw);
19688:ToolBar2', 'TToolBar', iptrw);
19689: BtnOpen', 'TToolButton', iptrw);
19690: BtnSave', 'TToolButton', iptrw);
19691: BtnPrint', 'TToolButton', iptrw);
19692: BtnColors', 'TToolButton', iptrw);
19693: btnClassReport', 'TToolButton', iptrw);
19694: BtnRotateRight', 'TToolButton', iptrw);
19695: BtnFullScreen', 'TToolButton', iptrw);
19696: BtnFitToWindowSize', 'TToolButton', iptrw);
19697: BtnZoomMinus', 'TToolButton', iptrw);
19698: BtnZoomPlus', 'TToolButton', iptrw);
19699: Panel1', 'TPanel', iptrw);
19700: LabelBrettgroesse', ' TLabel', iptrw);
19701: CBLSCList', 'TComboBox', iptrw);
19702: ImageListNormal', 'TImageList', iptrw);
19703: spbtexplore', 'TSpeedButton', iptrw);
19704: spbtnexampel', 'TSpeedButton', iptrw);
19705: spbsaveas', 'TSpeedButton', iptrw);
19706: imglogobox', 'TImage', iptrw);
19707: EnlargeFont1', 'TMenuItem', iptrw);
19708: EnlargeFont2', 'TMenuItem', iptrw);
19709: ShrinkFont1', 'TMenuItem', iptrw);
19710: ThreadDemo1', 'TMenuItem', iptrw);
19711: HEXEditor1', 'TMenuItem', iptrw);
19712: HEXView1', 'TMenuItem', iptrw);
19713: HEXInspect1', 'TMenuItem', iptrw);
19714: SynExporterHTML1', 'TSynExporterHTML', iptrw);
19715: ExporttoHTML1', 'TMenuItem', iptrw);
19716: ClassCount1', 'TMenuItem', iptrw);
19717: HTMLOutput1', 'TMenuItem', iptrw);
19718: HEXEditor2', 'TMenuItem', iptrw);
19719: Minesweeper1', 'TMenuItem', iptrw);
19720: N17', 'TMenuItem', iptrw);
19721: PicturePuzzle1', 'TMenuItem', iptrw);
19722: sbvc1help', 'TSpeedButton', iptrw);
19723: DependencyWalker1', 'TMenuItem', iptrw);
19724: WebScanner1', 'TMenuItem', iptrw);
19725: View1', 'TMenuItem', iptrw);
19726: mnToolbar1', 'TMenuItem', iptrw);
19727: mnStatusbar2', 'TMenuItem', iptrw);
19728: mnConsole2', 'TMenuItem', iptrw);
19729: mnCoolbar2', 'TMenuItem', iptrw);
19730: mnSplitter2', 'TMenuItem', iptrw);
19731: WebServer1', 'TMenuItem', iptrw);
19732: Tutorial17Server1', 'TMenuItem', iptrw);
19733: Tutorial18Arduinol', 'TMenuItem', iptrw);
19734: SynPerlSyn1', 'TSynPerlSyn', iptrw);
19735: PerlSyntax1', 'TMenuItem', iptrw);
19736: SynPythonSyn1', 'TSynPythonSyn', iptrw);
19737: PythonSyntax1', 'TMenuItem', iptrw);
19738: DMathLibrary1', 'TMenuItem', iptrw);
19739: IntfNavigator1', 'TMenuItem', iptrw);
19740: EnlargeFontConsole1', 'TMenuItem', iptrw);
19741: ShrinkFontConsole1', 'TMenuItem', iptrw);
19742: SetInterfaceList1', 'TMenuItem', iptrw);
19743: popintfList', 'TPopupMenu', iptrw);
19744: intfAdd1', 'TMenuItem', iptrw);
19745: intfDelete1', 'TMenuItem', iptrw);
19746: intfRefactor1', 'TMenuItem', iptrw);
19747: Defactor1', 'TMenuItem', iptrw);
19748: Tutorial19COMArduinol', 'TMenuItem', iptrw);
19749: Tutorial20Regex', 'TMenuItem', iptrw);
```

```

19750: N18', 'TMenuItem', iptrw);
19751: ManualE1', 'TMenuItem', iptrw);
19752: FullTextFinder1', 'TMenuItem', iptrw);
19753: Move1', 'TMenuItem', iptrw);
19754: FractalDemo1', 'TMenuItem', iptrw);
19755: Tutorial21Android1', 'TMenuItem', iptrw);
19756: Tutorial0Function1', 'TMenuItem', iptrw);
19757: SimuLogBox1', 'TMenuItem', iptrw);
19758: OpenExamples1', 'TMenuItem', iptrw);
19759: SynJScriptSyn1', 'TSynJScriptSyn', iptrw);
19760: JavaScriptSyntax1', 'TMenuItem', iptrw);
19761: Halt1', 'TMenuItem', iptrw);
19762: CodeSearch1', 'TMenuItem', iptrw);
19763: SynRubySyn1', 'TSynRubySyn', iptrw);
19764: RubySyntax1', 'TMenuItem', iptrw);
19765: Undo1', 'TMenuItem', iptrw);
19766: SynUNIXShellsScriptSyn1', 'TSynUNIXShellScriptSyn', iptrw);
19767: LinuxShellScript1', 'TMenuItem', iptrw);
19768: Rename1', 'TMenuItem', iptrw);
19769: spdcodesearch', 'TSpeedButton', iptrw);
19770: Preview1', 'TMenuItem', iptrw);
19771: Tutorial22Services1', 'TMenuItem', iptrw);
19772: Tutorial23RealTime1', 'TMenuItem', iptrw);
19773: Configuration1', 'TMenuItem', iptrw);
19774: MP3Player1', 'TMenuItem', iptrw);
19775: DLLSpy1', 'TMenuItem', iptrw);
19776: SynURIOpener1', 'TSynURIOpener', iptrw);
19777: SynURISyn1', 'TSynURISyn', iptrw);
19778: URILinksClicks1', 'TMenuItem', iptrw);
19779: EditReplace1', 'TMenuItem', iptrw);
19780: GotoLine1', 'TMenuItem', iptrw);
19781: ActiveLineColor1', 'TMenuItem', iptrw);
19782: ConfigFile1', 'TMenuItem', iptrw);
19783: Sort1Intflist', 'TMenuItem', iptrw);
19784: Redo1', 'TMenuItem', iptrw);
19785: Tutorial24CleanCode1', 'TMenuItem', iptrw);
19786: Tutorial25Configuration1', 'TMenuItem', iptrw);
19787: IndentSelection1', 'TMenuItem', iptrw);
19788: UnindentSection1', 'TMenuItem', iptrw);
19789: SkyStyle1', 'TMenuItem', iptrw);
19790: N19', 'TMenuItem', iptrw);
19791: CountWords1', 'TMenuItem', iptrw);
19792: imbookmarkimages', 'TImageList', iptrw);
19793: Bookmark11', 'TMenuItem', iptrw);
19794: N20', 'TMenuItem', iptrw);
19795: Bookmark21', 'TMenuItem', iptrw);
19796: Bookmark31', 'TMenuItem', iptrw);
19797: Bookmark41', 'TMenuItem', iptrw);
19798: SynMultiSyn1', 'TSynMultiSyn', iptrw);
19799:

19800: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
19801: Procedure IFPS3ClassesPlugin1ExecImport(Sender: TObject; Exec: TPSExec; x:TPSRuntimeClassImporter);
19802: Procedure PSScriptCompile( Sender : TPSScript)
19803: Procedure Compile1Click( Sender : TObject)
19804: Procedure PSScriptExecute( Sender : TPSScript)
19805: Procedure open1Click( Sender : TObject)
19806: Procedure Save2Click( Sender : TObject)
19807: Procedure Savebefore1Click( Sender : TObject)
19808: Procedure Largefont1Click( Sender : TObject)
19809: Procedure FormActivate( Sender : TObject)
19810: Procedure SBytecode1Click( Sender : TObject)
19811: Procedure FormKeyPress( Sender : TObject; var Key : Char)
19812: Procedure Saveas3Click( Sender : TObject)
19813: Procedure Clear1Click( Sender : TObject)
19814: Procedure Slinenumbers1Click( Sender : TObject)
19815: Procedure About1Click( Sender : TObject)
19816: Procedure Search1Click( Sender : TObject)
19817: Procedure FormCreate( Sender : TObject)
19818: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Integer;
19819:                                     var Action : TSynReplaceAction)
19820: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
19821: Procedure WordWrap1Click( Sender : TObject)
19822: Procedure SearchNext1Click( Sender : TObject)
19823: Procedure Replace1Click( Sender : TObject)
19824: Function PSScriptNeedFile(Sender:TObject;const OriginFileName:String;var FName,Output:String):Bool;
19825: Procedure ShowInclude1Click( Sender : TObject)
19826: Procedure Printout1Click( Sender : TObject)
19827: Procedure mnuPrintFont1Click( Sender : TObject)
19828: Procedure Include1Click( Sender : TObject)
19829: Procedure FormDestroy( Sender : TObject)
19830: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
19831: Procedure UpdateView1Click( Sender : TObject)
19832: Procedure CodeCompletionList1Click( Sender : TObject)
19833: Procedure SaveOutput1Click( Sender : TObject)
19834: Procedure ExportClipboard1Click( Sender : TObject)
19835: Procedure Close1Click( Sender : TObject)
19836: Procedure Manual1Click( Sender : TObject)
19837: Procedure LoadLastFile1Click( Sender : TObject)
19838: Procedure Memo1Change( Sender : TObject)

```

```

19839: Procedure Decompile1Click( Sender : TObject )
19840: Procedure StepInTo1Click( Sender : TObject )
19841: Procedure StepOut1Click( Sender : TObject )
19842: Procedure Reset1Click( Sender : TObject )
19843: Procedure cedebbugAfterExecute( Sender : TPSScript )
19844: Procedure cedebbugBreakpoint(Sender:TObject; const FileName:String; Position,Row, Col: Cardinal)
19845: Procedure cedebbugCompile( Sender : TPSScript )
19846: Procedure cedebbugExecute( Sender : TPSScript )
19847: Procedure cedebbugIdle( Sender : TObject )
19848: Procedure cedebbugLineInfo( Sender:TObject;const FileName:String; Position, Row, Col : Cardinal )
19849: Procedure Mem10SpecialLineColors(Sender: TObject; Line:Int; var Special:Boolean;var FG,BG:TColor );
19850: Procedure BreakPointMenuItemClick( Sender : TObject )
19851: Procedure DebugRun1Click( Sender : TObject )
19852: Procedure tutorial4Click( Sender : TObject )
19853: Procedure ImportfromClipboard1Click( Sender : TObject )
19854: Procedure ImportfromClipboard2Click( Sender : TObject )
19855: Procedure tutorial1Click( Sender : TObject )
19856: Procedure ShowSpecChars1Click( Sender : TObject )
19857: Procedure StatusBar1DblClick( Sender : TObject )
19858: Procedure PSScriptLine( Sender : TObject )
19859: Procedure OpenDirectory1Click( Sender : TObject )
19860: Procedure procMessClick( Sender : TObject )
19861: Procedure tbtnUseCaseClick( Sender : TObject )
19862: Procedure EditFont1Click( Sender : TObject )
19863: Procedure tutorial21Click( Sender : TObject )
19864: Procedure tutorial31Click( Sender : TObject )
19865: Procedure HTMLSyntax1Click( Sender : TObject )
19866: Procedure ShowInterfaces1Click( Sender : TObject )
19867: Procedure Tutorial5Click( Sender : TObject )
19868: Procedure ShowLastException1Click( Sender : TObject )
19869: Procedure PlayMP31Click( Sender : TObject )
19870: Procedure AllFunctionsList1Click( Sender : TObject )
19871: Procedure texSyntax1Click( Sender : TObject )
19872: Procedure GetEMails1Click( Sender : TObject )
19873: procedure DelphiSite1Click(Sender: TObject);
19874: procedure TerminalStyle1Click(Sender: TObject);
19875: procedure ReadOnly1Click(Sender: TObject);
19876: procedure ShellStyle1Click(Sender: TObject);
19877: procedure Console1Click(Sender: TObject); //3.2
19878: procedure BigScreen1Click(Sender: TObject);
19879: procedure Tutorial91Click(Sender: TObject);
19880: procedure SaveScreenshotClick(Sender: TObject);
19881: procedure Tutorial101Click(Sender: TObject);
19882: procedure SQLSyntax1Click(Sender: TObject);
19883: procedure XMLSyntax1Click(Sender: TObject);
19884: procedure ComponentCount1Click(Sender: TObject);
19885: procedure NewInstance1Click(Sender: TObject);
19886: procedure CSyntax1Click(Sender: TObject);
19887: procedure Tutorial6Click(Sender: TObject);
19888: procedure New1Click(Sender: TObject);
19889: procedure AllObjectsList1Click(Sender: TObject);
19890: procedure LoadBytecode1Click(Sender: TObject);
19891: procedure CipherFile1Click(Sender: TObject); //V3.5
19892: procedure NewInstance1Click(Sender: TObject);
19893: procedure toolbtnTutorialClick(Sender: TObject);
19894: procedure Memory1Click(Sender: TObject);
19895: procedure JavaSyntax1Click(Sender: TObject);
19896: procedure SyntaxCheck1Click(Sender: TObject);
19897: procedure ScriptExplorer1Click(Sender: TObject);
19898: procedure FormOutput1Click(Sender: TObject); //V3.6
19899: procedure GotoEnd1Click(Sender: TObject);
19900: procedure AllResourceList1Click(Sender: TObject);
19901: procedure tbtn6resClick(Sender: TObject); //V3.7
19902: procedure Info1Click(Sender: TObject);
19903: procedure Tutorial10Statistics1Click(Sender: TObject);
19904: procedure Tutorial11Forms1Click(Sender: TObject);
19905: procedure Tutorial12SQL1Click(Sender: TObject); //V3.8
19906: procedure ResourceExplore1Click(Sender: TObject);
19907: procedure InfolClick(Sender: TObject);
19908: procedure CryptoBox1Click(Sender: TObject);
19909: procedure ModulesCount1Click(Sender: TObject);
19910: procedure N4GewinntGame1Click(Sender: TObject);
19911: procedure PHPSyntax1Click(Sender: TObject);
19912: procedure SerialRS2321Click(Sender: TObject);
19913: procedure CSyntax2Click(Sender: TObject);
19914: procedure Calculator1Click(Sender: TObject);
19915: procedure Tutorial13Ciphering1Click(Sender: TObject);
19916: procedure Tutorial14Async1Click(Sender: TObject);
19917: procedure PHPSyntax1Click(Sender: TObject);
19918: procedure BtnZoomPlusClick(Sender: TObject);
19919: procedure BtnZoomMinusClick(Sender: TObject);
19920: procedure btnClassReportClick(Sender: TObject);
19921: procedure ThreadDemo1Click(Sender: TObject);
19922: procedure HEXView1Click(Sender: TObject);
19923: procedure ExporttoHTML1Click(Sender: TObject);
19924: procedure Minesweeper1Click(Sender: TObject);
19925: procedure PicturePuzzle1Click(Sender: TObject); //V3.9
19926: procedure sbvc1helpClick(Sender: TObject);
19927: procedure DependencyWalker1Click(Sender: TObject);

```

```

19928: procedure CB1SCListDrawItem(Control:TWinControl;Index:Int;aRect:TRect;State:TOwnerDrawState);
19929: procedure WebScanner1Click(Sender: TObject);
19930: procedure mnToolbar1Click(Sender: TObject);
19931: procedure mnStatusbar2Click(Sender: TObject);
19932: procedure mnConsole2Click(Sender: TObject);
19933: procedure mnCoolbar2Click(Sender: TObject);
19934: procedure mnSplitter2Click(Sender: TObject);
19935: procedure WebServer1Click(Sender: TObject);
19936: procedure PerlSyntax1Click(Sender: TObject);
19937: procedure PythonSyntax1Click(Sender: TObject);
19938: procedure DMathLibrary1Click(Sender: TObject);
19939: procedure IntfNavigator1Click(Sender: TObject);
19940: procedure FullTextFinder1Click(Sender: TObject);
19941: function AppName: string;
19942: function ScriptName: string;
19943: function LastName: string;
19944: procedure FractalDemo1Click(Sender: TObject);
19945: procedure SimulLogBox1Click(Sender: TObject);
19946: procedure OpenExamples1Click(Sender: TObject);
19947: procedure Halt1Click(Sender: TObject);
19948: procedure Stop;
19949: procedure CodeSearch1Click(Sender: TObject);
19950: procedure RubySyntax1Click(Sender: TObject);
19951: procedure Undo1Click(Sender: TObject);
19952: procedure LinuxShellScript1Click(Sender: TObject);
19953: procedure WebScannerDirect(urls: string);
19954: procedure WebScanner(urls: string);
19955: procedure LoadInterfaceList2;
19956: procedure DLLSpy1Click(Sender: TObject);
19957: procedure Memo1Db1Click(Sender: TObject);
19958: procedure URILinksClicks1Click(Sender: TObject);
19959: procedure GotoLine1Click(Sender: TObject);
19960: procedure ConfigFile1Click(Sender: TObject);
19961: Procedure Sort1IntflistClick( Sender : TObject )
19962: Procedure Redo1Click( Sender : TObject )
19963: Procedure Tutorial24CleanCode1Click( Sender : TObject )
19964: Procedure IndentSelection1Click( Sender : TObject )
19965: Procedure UnindentSection1Click( Sender : TObject )
19966: Procedure SkyStyle1Click( Sender : TObject )
19967: Procedure CountWords1Click( Sender : TObject )
19968: Procedure Memo1PlaceBookmark( Sender : TObject; var Mark : TSynEditMark )
19969: Procedure Memo1GutterClick(Sender:TObject;Button:TMouseButton;X,Y,Line:Integer;Mark:TSynEditMark);
19970: Procedure Bookmark11Click( Sender : TObject );
19971: Procedure Bookmark21Click( Sender : TObject );
19972: Procedure Bookmark31Click( Sender : TObject );
19973: Procedure Bookmark41Click( Sender : TObject );
19974: Procedure SynMultiSyn1CustomRange(Sender:TSynMultiSyn;Operat:TRangeOperation;var Range:Pointer);
19975: 'STATMemoryReport', 'boolean', iptrw);
19976: 'IPPort', 'integer', iptrw);
19977: 'COMPort', 'integer', iptrw);
19978: 'lbintlist', 'TListBox', iptrw);
19979: Function GetStatChange : boolean
19980: Procedure SetStatChange( vstat : boolean )
19981: Function GetActFileName : string
19982: Procedure SetActFileName( vname : string )
19983: Function GetLastFileName : string
19984: Procedure SetLastFileName( vname : string )
19985: Procedure WebScannerDirect( urls : string )
19986: Procedure LoadInterfaceList2
19987: Function GetStatExecuteShell : boolean
19988: Procedure DoEditorExecuteCommand( EditorCommand : word )
19989: function GetActiveLineColor: TColor
19990: procedure SetActiveLineColor(acolor: TColor)
19991: procedure ScriptListbox1Click(Sender: TObject);
19992: procedure Memo2KeyPress(Sender: TObject; var Key: Char);
19993: procedure EnlargeGutter1Click(Sender: TObject);
19994: procedure Tetris1Click(Sender: TObject);
19995: procedure ToDoList1Click(Sender: TObject);
19996: procedure ProcessList1Click(Sender: TObject);
19997: procedure MetricReport1Click(Sender: TObject);
19998: procedure ProcessList1Click(Sender: TObject);
19999: procedure TCPSockets1Click(Sender: TObject);
20000: procedure ConfigUpdate1Click(Sender: TObject);
20001: procedure ADOWorkbench1Click(Sender: TObject);
20002: procedure SocketServer1Click(Sender: TObject);
20003: procedure FormDemo1Click(Sender: TObject);
20004: procedure Richedit1Click(Sender: TObject);
20005: procedure SimpleBrowser1Click(Sender: TObject);
20006: procedure DOSShell1Click(Sender: TObject);
20007: procedure SynExport1Click(Sender: TObject);
20008: procedure ExporttoRTF1Click(Sender: TObject);
20009: procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
20010: procedure SOAPTester1Click(Sender: TObject);
20011: procedure Sniffer1Click(Sender: TObject);
20012: procedure AutoDetectSyntax1Click(Sender: TObject);
20013: procedure FPlot1Click(Sender: TObject);
20014: procedure PasStyle1Click(Sender: TObject);
20015: procedure Tutorial183RGBLED1Click(Sender: TObject);
20016: procedure Reversi1Click(Sender: TObject);

```

```

20017: procedure ManualmaxBox1Click(Sender: TObject);
20018: procedure BlaisePascalMagazine1Click(Sender: TObject);
20019: procedure AddToDo1Click(Sender: TObject);
20020: procedure CreateGUID1Click(Sender: TObject);
20021: procedure Tutorial27XML1Click(Sender: TObject);
20022: procedure CreateDLLStub1Click(Sender: TObject);
20023: procedure Tutorial28DLL1Click(Sender: TObject);');
20024: procedure ResetKeyPressed;');
20025: procedure FileChanges1Click(Sender: TObject);');
20026: procedure OpenGLTry1Click(Sender: TObject);');
20027: procedure AllUnitList1Click(Sender: TObject);');
20028: procedure Tutorial29UMLClick(Sender: TObject);
20029: procedure CreateHeader1Click(Sender: TObject);
20030:
20031: //-----
20032: //*****mX4 Editor SynEdit Tools API *****
20033: //-----
20034: procedure SIRegister_TCustomSynEdit(CL: TSPSPascalCompiler);
20035: begin
20036:   //with RegClassS(CL,'TCustomControl', 'TCustomSynEdit') do
20037:   with FindClass('TCustomControl','TCustomSynEdit') do begin
20038:     Constructor Create( AOwner : TComponent )
20039:       SelStart', 'Integer', iptrw);
20040:       SelEnd', 'Integer', iptrw); AlwaysShowCaret', 'Boolean', iptrw);
20041:     Procedure UpdateCaret
20042:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2:word;SS2:TShiftState);
20043:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2: word;SS2:TShiftState);
20044:     Procedure BeginUndoBlock
20045:     Procedure BeginUpdate
20046:     Function CaretInView : Boolean
20047:     Function CharIndexToRowCol( Index : integer ) : TBufferCoord
20048:     Procedure Clear
20049:     Procedure ClearAll
20050:     Procedure ClearBookMark( BookMark : Integer )
20051:     Procedure ClearSelection
20052:     Procedure CommandProcessor( Command : TSynEditorCommand; AChar : char; Data : pointer )
20053:     Procedure ClearUndo
20054:     Procedure CopyToClipboard
20055:     Procedure CutToClipboard
20056:     Procedure DoCopyToClipboard( const SText : string )
20057:     Procedure EndUndoBlock
20058:     Procedure EndUpdate
20059:     Procedure EnsureCursorPosVisible
20060:     Procedure EnsureCursorPosVisibleEx( ForceToMiddle : Boolean )
20061:     Procedure FindMatchingBracket
20062:     Function GetMatchingBracket : TBufferCoord
20063:     Function GetMatchingBracketEx( const APoint : TBufferCoord ) : TBufferCoord
20064:     Procedure ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer )
20065:     Function GetBookMark( BookMark : integer; var X, Y : integer ) : boolean
20066:     Function GetHighlighterAttriAtRowCol( const XY : TBufferCoord; var Token : string; var Attr
20067:       : TSynHighlighterAttributes ) : boolean
20068:     Function GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token : string;
20069:       var TokenType, Start : Integer; var Attr:TSynHighlighterAttributes):boolean
20070:     Function GetPositionOfMouse( out aPos : TBufferCoord ) : Boolean
20071:     Function GetWordAtRowCol( const XY : TBufferCoord ) : string
20072:     Procedure GotoBookMark( BookMark : Integer )
20073:     Procedure GotoLineAndCenter( ALine : Integer )
20074:     Function IdentChars : TSynIdentChars
20075:     Procedure InvalidateGutter
20076:     Procedure InvalidateGutterLine( aLine : integer )
20077:     Procedure InvalidateGutterLines( FirstLine, LastLine : integer )
20078:     Procedure InvalidateLine( Line : integer )
20079:     Procedure InvalidateLines( FirstLine, LastLine : integer )
20080:     Procedure InvalidateSelection
20081:     Function IsBookmark( BookMark : integer ) : boolean
20082:     Function IsPointInSelection( const Value : TBufferCoord ) : boolean
20083:     Procedure LockUndo
20084:     Function BufferToDisplayPos( const p : TBufferCoord ) : TDisplayCoord
20085:     Function DisplayToBufferPos( const p : TDisplayCoord ) : TBufferCoord
20086:     Function LineToRow( aLine : integer ) : integer
20087:     Function RowToLine( aRow : integer ) : integer
20088:     Function NextWordPos : TBufferCoord
20089:     Function NextWordPosEx( const XY : TBufferCoord ) : TBufferCoord
20090:     Procedure PasteFromClipboard
20091:     Function WordStart : TBufferCoord
20092:     Function WordStartEx( const XY : TBufferCoord ) : TBufferCoord
20093:     Function WordEnd : TBufferCoord
20094:     Function WordEndEx( const XY : TBufferCoord ) : TBufferCoord
20095:     Function PrevWordPos : TBufferCoord
20096:     Function PrevWordPosEx( const XY : TBufferCoord ) : TBufferCoord
20097:     Function PixelsToRowColumn( aX, aY : integer ) : TDisplayCoord
20098:     Function PixelsToNearestRowColumn( aX, aY : integer ) : TDisplayCoord
20099:     Procedure Redo
20100:     Procedure RegisterCommandHandler(const AHandlerProc:THookedCommandEvent;AHandlerData:pointer);
20101:     Function RowColumnToPixels( const RowCol : TDisplayCoord ) : TPoint
20102:     Function RowColToCharIndex( RowCol : TBufferCoord ) : integer
20103:     Function SearchReplace( const ASearch, AReplace:string; AOptions:TSynSearchOptions): integer
20104:     Procedure SelectAll
20105:     Procedure SetBookMark( BookMark : Integer; X : Integer; Y : Integer )

```

```

20106: Procedure SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord)
20107: Procedure SetDefaultKeystrokes
20108: Procedure SetSelWord
20109: Procedure SetWordBlock( Value : TBufferCoord)
20110: Procedure Undo
20111: Procedure UnlockUndo
20112: Procedure UnregisterCommandHandler( AHandlerProc : THookedCommandEvent )
20113: Procedure AddKeyUpHandler( aHandler : TKeyEvent )
20114: Procedure RemoveKeyUpHandler( aHandler : TKeyEvent )
20115: Procedure AddKeyDownHandler( aHandler : TKeyEvent )
20116: Procedure RemoveKeyDownHandler( aHandler : TKeyEvent )
20117: Procedure AddKeyPressHandler( aHandler : TKeyPressEvent )
20118: Procedure RemoveKeyPressHandler( aHandler : TKeyPressEvent )
20119: Procedure AddFocusControl( aControl : TWinControl )
20120: Procedure RemoveFocusControl( aControl : TWinControl )
20121: Procedure AddMouseDownHandler( aHandler : TMouseEvent )
20122: Procedure RemoveMouseDownHandler( aHandler : TMouseEvent )
20123: Procedure AddMouseUpHandler( aHandler : TMouseEvent )
20124: Procedure RemoveMouseUpHandler( aHandler : TMouseEvent )
20125: Procedure AddMouseCursorHandler( aHandler : TMouseEvent )
20126: Procedure RemoveMouseCursorHandler( aHandler : TMouseEvent )
20127: Procedure SetLinesPointer( ASynEdit : TCustomSynEdit )
20128: Procedure RemoveLinesPointer
20129: Procedure HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList )
20130: Procedure UnHookTextBuffer
20131: BlockBegin', 'TBufferCoord', iptrw);
20132: BlockEnd', 'TBufferCoord', iptrw);
20133: CanPaste', 'Boolean', iptr);
20134: CanRedo', 'boolean', iptr);
20135: CanUndo', 'boolean', iptr);
20136: CaretX', 'Integer', iptrw);
20137: CaretY', 'Integer', iptrw);
20138: CaretXY', 'TBufferCoord', iptrw);
20139: ActiveLineColor', 'TColor', iptrw);
20140: DisplayX', 'Integer', iptr);
20141: DisplayY', 'Integer', iptr);
20142: DisplayXY', 'TDisplayCoord', iptr);
20143: DisplayLineCount', 'integer', iptr);
20144: CharsInWindow', 'Integer', iptr);
20145: CharWidth', 'integer', iptr);
20146: Font', 'TFont', iptrw);
20147: GutterWidth', 'Integer', iptr);
20148: Highlighter', 'TSynCustomHighlighter', iptrw);
20149: LeftChar', 'Integer', iptrw);
20150: LineHeight', 'integer', iptr);
20151: LinesInWindow', 'Integer', iptr);
20152: LineText', 'string', iptrw);
20153: Lines', 'TStrings', iptrw);
20154: Marks', 'TSynEditMarkList', iptr);
20155: MaxScrollWidth', 'integer', iptrw);
20156: Modified', 'Boolean', iptrw);
20157: PaintLock', 'Integer', iptr);
20158: ReadOnly', 'Boolean', iptrw);
20159: SearchEngine', 'TSynEditSearchCustom', iptrw);
20160: SelAvail', 'Boolean', iptr); SelLength', 'integer', iptrw);
20161: SelTabBlock', 'Boolean', iptr);
20162: SelTabLine', 'Boolean', iptr);
20163: SelText', 'string', iptrw);
20164: StateFlags', 'TSynStateFlags', iptr);
20165: Text', 'string', iptrw);
20166: TopLine', 'Integer', iptrw);
20167: WordAtCursor', 'string', iptr);
20168: WordAtMouse', 'string', iptr);
20169: UndoList', 'TSynEditUndoList', iptr);
20170: RedoList', 'TSynEditUndoList', iptr);
20171: OnProcessCommand', 'TProcessCommandEvent', iptrw);
20172: BookMarkOptions', 'TSynBookMarkOpt', iptrw);
20173: BorderStyle', 'TSynBorderStyle', iptrw);
20174: ExtraLineSpacing', 'integer', iptrw);
20175: Gutter', 'TSynGutter', iptrw);
20176: HideSelection', 'boolean', iptrw);
20177: InsertCaret', 'TSynEditCaretType', iptrw);
20178: InsertMode', 'boolean', iptrw);
20179: IsScrolling', 'Boolean', iptr);
20180: Keystrokes', 'TSynEditKeyStrokes', iptrw);
20181: MaxUndo', 'Integer', iptrw);
20182: Options', 'TSynEditorOptions', iptrw);
20183: OverwriteCaret', 'TSynEditCaretType', iptrw);
20184: RightEdge', 'Integer', iptrw); RightEdgeColor', 'TColor', iptrw);
20185: ScrollHintColor', 'TColor', iptrw);
20186: ScrollHintFormat', 'TScrollHintFormat', iptrw);
20187: ScrollBars', 'TScrollStyle', iptrw);
20188: SelectedColor', 'TSynSelectedColor', iptrw);
20189: SelectionMode', 'TSynSelectionMode', iptrw);
20190: ActiveSelectionMode', 'TSynSelectionMode', iptrw);
20191: TabWidth', 'integer', iptrw); WantReturns', 'boolean', iptrw);
20192: WantTabs', 'boolean', iptrw); WordWrap', 'boolean', iptrw);
20193: WordWrapGlyph', 'TSynGlyph', iptrw);
20194: OnChange', 'TNotifyEvent', iptrw);

```

```

20195:     OnClearBookmark', 'TPlaceMarkEvent', iptrw);
20196:     OnCommandProcessed', 'TProcessCommandEvent', iptrw);
20197:     OnContextHelp', 'TContextHelpEvent', iptrw);
20198:     OnDropFiles', 'TDropFilesEvent', iptrw);
20199:     OnGutterClick', 'TGutterClickEvent', iptrw);
20200:     OnGutterGetText', 'TGutterGetTextEvent', iptrw);
20201:     OnGutterPaint', 'TGutterPaintEvent', iptrw);
20202:     OnMouseCursor', 'TMouseCursorEvent', iptrw);
20203:     OnPaint', 'TPaintEvent', iptrw);
20204:     OnPlaceBookmark', 'TPlaceMarkEvent', iptrw);
20205:     OnProcessUserCommand', 'TProcessCommandEvent', iptrw);
20206:     OnReplaceText', 'TReplaceTextEvent', iptrw);
20207:     OnSpecialLineColors', 'TSpecialLineColorsEvent', iptrw);
20208:     OnStatusChange', 'TStatusChangeEvent', iptrw);
20209:     OnPaintTransient', 'TPaintTransient', iptrw);
20210:     OnScroll', 'TScrollEvent', iptrw);
20211:   end;
20212: Procedure RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)
20213: Function GetPlaceableHighlighters : TSynHighlighterList
20214: Function EditorCommandToDescrString( Cmd : TSynEditorCommand ) : string
20215: Function EditorCommandToCodeString( Cmd : TSynEditorCommand ) : string
20216: Procedure GetEditorCommandValues( Proc : TGetStrProc )
20217: Procedure GetEditorCommandExtended( Proc : TGetStrProc )
20218: Function IdentToEditorCommand( const Ident : string; var Cmd : longint ) : boolean
20219: Function EditorCommandToIdent( Cmd : longint; var Ident : string ) : boolean
20220: Function ConvertCodeStringToExtended( AString : String ) : String
20221: Function ConvertExtendedToCodeString( AString : String ) : String
20222: Function ConvertExtendedToCommand( AString : String ) : TSynEditorCommand
20223: Function ConvertCodeStringToCommand( AString : String ) : TSynEditorCommand
20224: Function IndexToEditorCommand( const AIndex : Integer ) : Integer
20225:
20226: TSynEditorOption =
20227:   eoAltSetsColumnMode,           //Holding down the Alt Key will put the selection mode into columnar format
20228:   eoAutoIndent,                //Will indent caret on newlines with same amount of leading whitespace as
20229:                           //preceding line
20230:   eoAutoSizeMaxScrollWidth,    //Automatically resizes the MaxScrollWidth property when inserting text
20231:   eoDisableScrollArrows,       //Disables the scroll bar arrow buttons when you can't scroll in that
20232:                           //direction any more
20233:   eoDragDropEditing,          //Allows to select a block of text and drag it within document to another
20234:                           //location
20235:   eoDropFiles,                 //Allows the editor accept OLE file drops
20236:   eoEnhanceHomeKey,           //enhances home key positioning, similar to visual studio
20237:   eoEnhanceEndKey,            //enhances End key positioning, similar to JDeveloper
20238:   eoGroupUndo,                 //When undoing/redoing actions, handle all continuous changes the same kind
20239:                           //in one call
20240:                           //instead undoing/redoing each command separately
20241:   eoHalfPageScroll,           //When scrolling with page-up and page-down commands, only scroll a half
20242:                           //page at a time
20243:   eoHideShowScrollbars,        //if enabled, then scrollbars will only show if necessary.
20244: If you have ScrollPasteEOL, then it the horizontal bar will always be there (it uses MaxLength instead)
20245:   eoKeepCaretX,               //When moving through lines w/o cursor Past EOL, keeps X position of cursor
20246:   eoNoCaret,                  //Makes it so the caret is never visible
20247:   eoNoSelection,              //Disables selecting text
20248:   eoRightMouseMovesCursor,    //When clicking with right mouse for popup menu, moves cursor to location
20249:   eoScrollByOneLess,          //Forces scrolling to be one less
20250:   eoScrollHintFollows,         //The scroll hint follows the mouse when scrolling vertically
20251:   eoScrollPastEof,            //Allows the cursor to go past the end of file marker
20252:   eoScrollPastEol,             //Allows cursor to go past last character into white space at end of a line
20253:   eoShowScrollHint,            //Shows a hint of the visible line numbers when scrolling vertically
20254:   eoShowSpecialChars,          //Shows the special Characters
20255:   eoSmartTabDelete,            //similar to Smart Tabs, but when you delete characters
20256:   eoSmartTabs,                 //When tabbing, cursor will go to non-white space character of previous line
20257:   eoSpecialLineDefaultFg,      //disables the foreground text color override using OnSpecialLineColor event
20258:   eoTabIndent,                  //If active <Tab>and<Shift><Tab> act block indent,unindent when text select
20259:   eoTabsToSpaces,              //Converts a tab character to a specified number of space characters
20260:   eoTrimTrailingSpaces        //Spaces at the end of lines will be trimmed and not saved
20261:
20262: *****Important Editor Short Cuts*****;
20263: Double click to select a word and count words with highlightning.
20264: Triple click to select a line.
20265: CTRL+SHIFT+click to extend a selection.
20266: Drag with the ALT key down to select columns of text !!!
20267: Drag and drop is supported.
20268: Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
20269: Type CTRL+A to select all.
20270: Type CTRL+N to set a new line.
20271: Type CTRL+T to delete a line or token. //Tokenizer
20272: Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
20273: Type CTRL+Shift+T to add ToDo in line and list.
20274: Type CTRL+Shift+[0..9] to set bookmarks. //Bookmark
20275: Type CTRL[0..9] to jump or get to bookmarks.
20276: Type Home to position cursor at beginning of current line and End to position it at end of line.
20277: Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of document.
20278: Page Up and Page Down work as expected.
20279: CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
20280: using http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
20281:
20282: {$ Short Key Positions Ctrl<A-Z>: }
20283: def

```

```

20284:   <A> Select All
20285:   <B> Count Words
20286:   <C> Copy
20287:   <D> Internet Start
20288:   <E> Script List
20289:   <F> Find
20290:   <G> Goto
20291:   <H> Mark Line
20292:   <I> Interface List
20293:   <J> Code Completion
20294:   <K> Console
20295:   <L> Interface List Box
20296:   <M> Font Larger -
20297:   <N> New Line
20298:   <O> Open File
20299:   <P> Font Smaller +
20300:   <Q> Quit
20301:   <R> Replace
20302:   <S> Save!
20303:   <T> Delete Line
20304:   <U> Use Case Editor
20305:   <V> Paste
20306:   <W> URI Links
20307:   <X> Reserved for coding use internal
20308:   <Y> Delete Line
20309:   <Z> Undo
20310:
20311: ref
20312:   F1 Help
20313:   F2 Syntax Check
20314:   F3 Search Next
20315:   F4 New Instance
20316:   F5 Line Mark /Breakpoint
20317:   F6 Goto End
20318:   F7 Debug Step Into
20319:   F8 Debug Step Out
20320:   F9 Compile
20321:   F10 Menu
20322:   F11 Word Count Highlight
20323:   F12 Reserved for coding use internal
20324:
20325: def ReservedWords: array[0..82] of string =
20326:   ('and', 'array', 'as', 'asm', 'at', 'begin', 'case', 'class', 'const',
20327:    'constructor', 'default', 'destructor', 'disinterface', 'div', 'do',
20328:    'downto', 'else', 'end', 'except', 'exports', 'file', 'finalization',
20329:    'finally', 'for', 'function', 'goto', 'if', 'implementation', 'in',
20330:    'inherited', 'initialization', 'inline', 'interface', 'is', 'label',
20331:    'library', 'message', 'mod', 'nil', 'not', 'object', 'of', 'on', 'or',
20332:    'out', 'packed', 'procedure', 'program', 'property', 'raise', 'read',
20333:    'record', 'repeat', 'resourcestring', 'set', 'shl', 'shr', 'string',
20334:    'stored', 'then', 'threadvar', 'to', 'try', 'type', 'unit', 'until',
20335:    'uses', 'var', 'while', 'with', 'write', 'xor', 'private', 'protected',
20336:    'public', 'published', 'def', 'ref', 'using', 'typedef', 'memo1', 'memo2', 'doc', 'maxform1';
20337: AllowedChars: array[0..5] of string = ('(', ')', '[', ']', ',', ',', t,t1,t2,t3: boolean;
20338: //-----
20339: //*****End of mx4 Public Tools API *****
20340: //-----
20341:
20342: Amount of Functions: 12983
20343: Amount of Procedures: 7997
20344: Amount of Constructors: 1292
20345: Totals of Calls: 22272
20346: SHA1: Win Exe 3.9.9.98 2B7D1EBFB07DB9B0660F857320C44D70CAA1C717
20347:
20348: ****
20349: Doc Short Manual with 50 Tips!
20350: ****
20351: - Install: just save your maxboxdef.ini before and then extract the zip file!
20352: - Toolbar: Click on the red maxbox Sign (right on top) opens your work directory or jump to <Help>
20353: - Menu: With <F2> you check syntax with <F8> you debug and <F9> you compile!
20354: - Menu: With <Ctrl><F3> you can search for code on examples
20355: - Menu: Open in menu Output a new instance <F4> of the box to compare or prepare your scripts
20356: - Menu: Set Interface Naviagator in menu /View/Intf Navigator
20357: - Menu: Switch or toogle between the last 2 scripts in menu File/LoadLast (History is set to 9 files)
20358:
20359: - Inifile: Set memory report in ini: MEMORYREPORT=Y :report on memory leaks on shutdown by dialog
20360: - Inifile: Refresh (reload) the inifile after edit with ..//Help/Config Update
20361: - Context Menu: You can printout your scripts as a pdf-file or html-export
20362: - Context: You do have a context menu with the right mouse click
20363:
20364: - Menu: With the UseCase Editor you can convert graphic formats too.
20365: - Menu: On menu Options you find Addons as compiled scripts
20366: - IDE: You don't need a mouse to handle maxbox, use shortcuts
20367: - Menu: Check Options/ProcessMessages! if something is wrong or you can't see graphics in a time
20368: - IDE: Dragndrop your scripts in box or the model in use case editor (Cut,Copy,Paste always available)
20369: - Editor: You can get templates as code completion with <ctrl j> in editor like classp or interface
20370:           or ttimer (you type classp and then CTRL J), or you type tstringlist and <Ctrl><J>
20371:
20372: - Menu: In menu output (console) you can set output menu in edit mode by unchecking <read only output>

```

```

20373: - Editor: After the end. you can write or copy notes or descriptions concerning the app or code
20374: - Code: If you code a loop till key-pressed use function: isKeyPressed;
20375: - Code: Macro set the macros #name, #paAdministrator, #file,startmaxbox_extract_funcList399.txter25.pdf
20376: - Code: change Syntax in autoboot macro 'maxbootscript.txt'  

20377: - Editor: - <F11> Click on Word in Editor search amount of words with highlighting, Dbl Click on Bookmarks  

20378:           to delete and Click and mark to drag a bookmark
20379: - Menu: To start handling from CD-ROM (read only mode) uncheck in Menu /Options/Save before Compile
20380: - IDE: A file info with system and script information you find in menu Program/Information
20381: - IDE: After change the config file in help you can update changes in menu Help/Config Update
20382: - IDE: Make a screenshot of the content and environment in menu Output/Save Screenshot
20383: - IDE: Use a boot loader script 'maxbootscript.txt' (as auto start) to change box each time you start it.
20384: - IDE: With escape or <Ctrl> Q you can also leave the box or stop a script in menu program - stop program
20385: - Editor: Set Bookmarks to check your work in app or code
20386: - Editor: With <Ctrl H> you set {SActive Line Color} and F11 you get Word Count Statistic on Output too
20387: - Editor: With //TODO: some description or DONE you set code entries for ToDo List in ..../Help/ToDo List
20388: - Editor: With <Ctrl W> you set active URL links in your code to test availability in Context Menu
20389:
20390: - IDE with menu /Options/ADO SQL Workbench you can manage your Database
20391: - Context Menu: You can write your docus with RichEdit RTF printout /Editor Form Options/Richedit
20392: - Menu: Set Interface Naviagator also with toggle <Ctrl L> or /View/Inft Navigator
20393: - Toolbar: In menu /View switch Off Toolbar and Coolbar to get more coding space
20394: - Code: Put some resources in your code /Help/Resource Explorer like bitbtn, forms, dialogs;
20395: - Code Editor: Compile with <F9> but also Alt C in case <F9> isn't available;
20396: - IDE set bookmarks with <Ctrl Shift> (0..9) and jump with <Ctrl> (0..9)
20397: - IDE menu /Help/Tools/ write with RTF Editor or open a DOS Shell or browse
20398:
20399: - Add on write your Requirements in RTF Docu with <CTRL ALT R> in context menu
20400: - Add on when no browser is available start /Options/Add ons/Easy Brower
20401: - Add on SOAP Tester with SOP POST File
20402: - Add on IP Protocol Sniffer with List View
20403: - Add on OpenGL mX Robot Demo for android
20404:
20405: - Menu: Help/Tools as a Tool Section with DOS Opener
20406: - Menu Editor: export the code as RTF File
20407: - Menu: Help/Tools/Syn Export your code as available in HTML or RTF
20408: - Menu: Help/Tools/ with <Alt H-S-D> you start the DOS Shell
20409: - Context: Auto Detect of Syntax depending on file extension
20410: - Code: some Windows API function start with w in the name like wGetAtomName();
20411: - IDE Close - if you cant close the box then reset it <Ctrl F2> in menu /Debug
20412: - IDE File Check with menu ..View/File Changes/...
20413: - Context: Create a Header with Create Header in Navigator List at right window
20414: - Code: use SysErrorMessage to get a real Error Description, Ex.
20415:           RemoveDir('c:\NoSuchFolder');
20416:           writeln('System Error Message: '+ SysErrorMessage(GetLastError));
20417: - IDE getWebScript from a URL with menu ..Help/Get Web Script/...
20418:
20419: - using DLL example in maxBox: //function: {*****}
20420:   Function GetProcessMemoryInfo(Process: THandle; var MemoryCounters: TProcessMemoryCounters;
20421:                                     cb: DWORD): BOOL; //stdcall;
20422:   External 'GetProcessMemoryInfo@psapi.dll stdcall';
20423:   Function OpenProcess(dwDesiredAccess:DWORD; bInheritHandle:BOOL; dwProcessId: DWORD):THHandle;
20424:   External 'OpenProcess@kernel32.dll stdcall';
20425:
20426: GCC Compile Ex Script
20427: procedure TFormMain_btnCompileClick(Sender: TObject);
20428: begin
20429:   AProcess:= TProcess.Create(Nil);
20430:   try
20431:     AProcess.CommandLine := 'gcc.exe "' + OpenDialog1.FileName + '"';
20432:     + ' -o "' + OpenDialog2.FileName + '"';
20433:     AProcess.Options:= AProcess.Options + [poWaitOnExit, poUsePipes];
20434:     AProcess.Execute;
20435:     Memo2.Lines.BeginUpdate;
20436:     Memo2.Lines.Clear;
20437:     Memo2.Lines.LoadFromStream(AProcess.Output);
20438:     Memo2.Lines.EndUpdate;
20439:   finally
20440:     AProcess.Free;
20441:   end;
20442: end;
20443:
20444: History Shell Hell
20445: PCT Precompile Technology , mX4 ScriptStudio
20446: Indy, JCL, Jedi, VCL, Systools, TurboPower, Fundamentals, ExtendedRTL, Synedit
20447: DMath, devC, Graphics32, ExtPascal, mX4, LCL, CLX, FCL, CPort and more
20448: emax layers: system-package-component-unit-class-function-block
20449: new keywords def ref using maXCalcF
20450: UML: use case act class state seq pac comp dep - lib lab
20451: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
20452: Tutorials, 30 Units add, VCL constructors, controls plus, unit list
20453: 2 Tutorials, 36 Units add, Synapse V40, LDAP, OpenSSL, AVScan
20454: 1 Report, 15 Units add, DBCtrls, Stream+, IRadio, Wininet
20455: 1 DLL Report, 24 Units add, DRTable, Remote+, Cindy functions!
20456: DLL Report, UML Tutor, 32 Units add, DRTable, Remote+, Cindy functions!
20457: Oscilloscope V4, Mixer, 17 Units add, URLMon, Form properties+, mathmax
20458: SendBuffer, Color+Caption hack, ComboSet, SetPrivilege, WakeOnLAN, ParaDice 3D Cube Polygraph, OCR
20459: GetScript or GetWebScript, GPS Example, Profiler, Checkers, Toolbox commons
20460: Add 15 Units, Wav resources, RoundTo, OpenOffice, Pipes
20461: TFixedCriticalSection, Xplatform beta, GCC Command Pipe

```

```

20462: Inno Install and Setup Routines
20463:
20464:
20465: https://unibe-ch.academia.edu/MaxKleiner
20466: www.slideshare.net/maxkleiner1
20467: http://www.scribd.com/max_kleiner
20468: http://www.delphiforfun.org/Programs/Utilities/index.htm
20469: http://www.slideshare.net/maxkleiner1
20470: http://s3.amazonaws.com/PreviewLinks/22959.html
20471: http://www.softwareschule.ch/arduino_training.pdf
20472: http://www.jrsoftware.org/isinfo.php
20473: http://www.be-precision.com/products/precision-builder/express/
20474:
20475:
20476:
20477:
20478: ****
20479: unit List asm internal end
20480: ****
20481: 01 unit RIRegister_Utils_Routines(exec); //Delphi
20482: 02 unit SIRegister_IdStrings //Indy Sockets
20483: 03 unit RIRegister_niSTRING_Routines(Exec); //from RegEx
20484: 04 unit uPSI_fMain Functions; //maXbox Open Tools API
20485: 05 unit IFSI_WinFormlpuzzle; //maXbox
20486: 06 unit RIRegister_LinarBitmap_Routines(Exec); //ImagefileLibBCB
20487: 07 unit RegisterDateTimeLibrary_R(exec); //Delphi
20488: 08 unit RIRegister_MathMax_Routines(exec); //Jedi & Delphi
20489: 09 unit RIRegister_IdGlobal_Routines(exec); //Indy Sockets
20490: 10 unit RIRegister_SysUtils_Routines(Exec); //Delphi
20491: 11 unit uPSI_IdTCPConnection; //Indy some functions
20492: 12 unit uPSCompiler.pas; //PS kernel functions
20493: 13 unit uPSI_DBCommon; //DB Common_Routines and Types
20494: 14 unit uPSI_Printers.pas //Delphi VCL
20495: 15 unit uPSI_MPlayer.pas //Delphi VCL
20496: 16 unit uPSC_comobj; //COM Functions
20497: 17 unit uPSI_Clipbrd; //Delphi VCL
20498: 18 unit Filectrl in IFSI_SysUtils_max; //VCL Runtime
20499: 19 unit uPSI_SqlExpr; //DBX3
20500: 20 unit uPSI_ADOdb; //ADODB
20501: 21 unit uPSI_StrHlpr; //String Helper Routines
20502: 22 unit uPSI_DateUtils; //Expansion to DateTimeLib
20503: 23 unit uPSI_FileUtils; //Expansion to Sys/File Utils
20504: 24 unit JUutils / gsUtils; //Jedi / Metabase
20505: 25 unit JvFunctions_max; //Jedi Functions
20506: 26 unit HTTPParser; //Delphi VCL
20507: 27 unit HTTPUtil; //Delphi VCL
20508: 28 unit uPSI_XMLUtil; //Delphi VCL
20509: 29 unit uPSI_SOAPHTTPClient; //Delphi VCL SOAP WebService V3.5
20510: 30 unit uPSI_Contrns; //Delphi RTL Container of Classes
20511: 31 unit uPSI_MaskUtils; //RTL Edit and Mask functions
20512: 32 unit uPSI_MyBigInt; //big integer class with Math
20513: 33 unit uPSI_ConvUtils; //Delphi VCL Conversions engine
20514: 34 unit Types_Variants; //Delphi\Win32\rtl\sys
20515: 35 unit uPSI_IdHashSHA1; //Indy Crypto Lib
20516: 36 unit uPSI_IdHashMessageDigest //Indy Crypto;
20517: 37 unit uPSI_IdASN1Util; //Indy ASN1Utility Routines;
20518: 38 unit uPSI_IdLogFile; //Indy Logger from LogBase
20519: 39 unit uPSI_IdICmpClient; //Indy Ping ICMP
20520: 40 unit uPSI_IdHashMessageDigest_max //Indy Crypto &OpenSSL;
20521: 41 unit uPSI_FileCtrl; //Delphi RTL
20522: 42 unit uPSI_Outline; //Delphi VCL
20523: 43 unit uPSI_ScktComp; //Delphi RTL
20524: 44 unit uPSI_Calendar; //Delphi VCL
20525: 45 unit uPSI_VListView; //VListView;
20526: 46 unit uPSI_DBGrids; //Delphi VCL
20527: 47 unit uPSI_DBCtrls; //Delphi VCL
20528: 48 unit ide_debugoutput; //maXbox
20529: 49 unit uPSI_ComCtrls; //Delphi VCL
20530: 50 unit uPSC_stdctrls+; //Delphi VCL
20531: 51 unit uPSI_Dialogs; //Delphi VCL
20532: 52 unit uPSI_StdConvs; //Delphi RTL
20533: 53 unit uPSI_DBClient; //Delphi RTL
20534: 54 unit uPSI_DBPlatform; //Delphi RTL
20535: 55 unit uPSI_Provider; //Delphi RTL
20536: 56 unit uPSI_FMTBcd; //Delphi RTL
20537: 57 unit uPSI_DBGrids; //Delphi VCL
20538: 58 unit uPSI_CDSUtil; //MIDAS
20539: 59 unit uPSI_VarHlpr; //Delphi RTL
20540: 60 unit uPSI_ExtDlgs; //Delphi VCL
20541: 61 unit sdpStopwatch; //maXbox
20542: 62 unit uPSI_JclStatistics; //JCL
20543: 63 unit uPSI_JclLogic; //JCL
20544: 64 unit uPSI_JclMiscel; //JCL
20545: 65 unit uPSI_JclMath_max; //JCL RTL
20546: 66 unit uPSI_uTPLb_StreamUtils; //LockBox 3
20547: 67 unit uPSI_MathUtils; //BCB
20548: 68 unit uPSI_JclMultimedia; //JCL
20549: 69 unit uPSI_WideStrUtils; //Delphi API/RTL
20550: 70 unit uPSI_GraphUtil; //Delphi RTL

```

```

20551: 71 unit uPSI_TypeTrans;                                //Delphi RTL
20552: 72 unit uPSI_HTTPApp;                               //Delphi VCL
20553: 73 unit uPSI_DBWeb;                                //Delphi VCL
20554: 74 unit uPSI_DBBeWeb;                               //Delphi VCL
20555: 75 unit uPSI_DBXpressWeb;                            //Delphi VCL
20556: 76 unit uPSI_ShadowWnd;                             //Delphi VCL
20557: 77 unit uPSI_ToolWin;                               //Delphi VCL
20558: 78 unit uPSI_Tabs;                                 //Delphi VCL
20559: 79 unit uPSI_JclGraphUtils;                         //JCL
20560: 80 unit uPSI_JclCounter;                            //JCL
20561: 81 unit uPSI_JclSysInfo;                            //JCL
20562: 82 unit uPSI_JclSecurity;                           //JCL
20563: 83 unit uPSI_JclFileUtils;                          //JCL
20564: 84 unit uPSI_IdUserAccounts;                        //Indy
20565: 85 unit uPSI_IdAuthentication;                      //Indy
20566: 86 unit uPSI_uTPLb_AES;                            //LockBox 3
20567: 87 unit uPSI_IdHashSHA1;                            //LockBox 3
20568: 88 unit uTPLb_BlockCipher;                           //LockBox 3
20569: 89 unit uPSI_ValEdit.pas;                           //Delphi VCL
20570: 90 unit uPSI_JvVCLUtils;                            //JCL
20571: 91 unit uPSI_JvDBUtil;                             //JCL
20572: 92 unit uPSI_JvDBUtils;                            //JCL
20573: 93 unit uPSI_JvAppUtils;                           //JCL
20574: 94 unit uPSI_JvCtrlUtils;                          //JCL
20575: 95 unit uPSI_JvFormToHtml;                          //JCL
20576: 96 unit uPSI_JvParsing;                            //JCL
20577: 97 unit uPSI_SerDlg;                              //Toolbox
20578: 98 unit uPSI_Serial;                             //Toolbox
20579: 99 unit uPSI_JvComponent;                          //JCL
20580: 100 unit uPSI_JvCalc;                             //JCL
20581: 101 unit uPSI_JvBdeutils;                          //JCL
20582: 102 unit uPSI_JvDateUtil;                          //JCL
20583: 103 unit uPSI_JvGenetic;                           //JCL
20584: 104 unit uPSI_JclBase;                            //JCL
20585: 105 unit uPSI_JvUtils;                            //JCL
20586: 106 unit uPSI_JvStrUtil;                           //JCL
20587: 107 unit uPSI_JvStrUtils;                          //JCL
20588: 108 unit uPSI_JvFileUtil;                           //JCL
20589: 109 unit uPSI_JvMemoryInfos;                      //JCL
20590: 110 unit uPSI_JvComputerInfo;                      //JCL
20591: 111 unit uPSI_JvgCommClasses;                     //JCL
20592: 112 unit uPSI_JvgLogics;                           //JCL
20593: 113 unit uPSI_JvLED;                             //JCL
20594: 114 unit uPSI_JvTurtle;                            //JCL
20595: 115 unit uPSI_SortThds; unit uPSI_ThSort;        //maxBox
20596: 116 unit uPSI_JvgUtils;                           //JCL
20597: 117 unit uPSI_JvExprParser;                        //JCL
20598: 118 unit uPSI_HexDump;                            //Borland
20599: 119 unit uPSI_DBLogDlg;                           //VCL
20600: 120 unit uPSI_SqlTimSt;                           //RTL
20601: 121 unit uPSI_JvHtmlParser;                        //JCL
20602: 122 unit uPSI_JvgXMLSerializer;                   //JCL
20603: 123 unit uPSI_JvJCLUtils;                          //JCL
20604: 124 unit uPSI_JvStrings;                           //JCL
20605: 125 unit uPSI_uTPLb_IntegerUtils;                  //TurboPower
20606: 126 unit uPSI_uTPLb_HugeCardinal;                 //TurboPower
20607: 127 unit uPSI_uTPLb_HugeCardinalUtils;            //TurboPower
20608: 128 unit uPSI_SynRegExpr;                          //SynEdit
20609: 129 unit uPSI_StUtils;                            //SysTools4
20610: 130 unit uPSI_StToHTML;                           //SysTools4
20611: 131 unit uPSI_StStrms;                           //SysTools4
20612: 132 unit uPSI_StFIN;                            //SysTools4
20613: 133 unit uPSI_StAstroP;                           //SysTools4
20614: 134 unit uPSI_StStat;                            //SysTools4
20615: 135 unit uPSI_StNetCon;                           //SysTools4
20616: 136 unit uPSI_StDecMth;                           //SysTools4
20617: 137 unit uPSI_StOStr;                            //SysTools4
20618: 138 unit uPSI_StPtrns;                           //SysTools4
20619: 139 unit uPSI_StNetMsg;                           //SysTools4
20620: 140 unit uPSI_StMath;                            //SysTools4
20621: 141 unit uPSI_StExpEng;                          //SysTools4
20622: 142 unit uPSI_StCRC;                            //SysTools4
20623: 143 unit uPSI_StExport;                           //SysTools4
20624: 144 unit uPSI_StExpLog;                          //SysTools4
20625: 145 unit uPSI_ActnList;                           //Delphi VCL
20626: 146 unit uPSI_jpeg;                             //Borland
20627: 147 unit uPSI_StRandom;                          //SysTools4
20628: 148 unit uPSI_StDict;                            //SysTools4
20629: 149 unit uPSI_StBCD;                            //SysTools4
20630: 150 unit uPSI_StTxtDat;                           //SysTools4
20631: 151 unit uPSI_StRegEx;                           //SysTools4
20632: 152 unit uPSI_IMouse;                            //VCL
20633: 153 unit uPSI_SyncObjs;                          //VCL
20634: 154 unit uPSI_AsyncCalls;                        //Hausladen
20635: 155 unit uPSI_ParallelJobs;                      //Saraiva
20636: 156 unit uPSI_Variants;                           //VCL
20637: 157 unit uPSI_VarCmplx;                          //VCL Wolfram
20638: 158 unit uPSI_DTDSchema;                         //VCL
20639: 159 unit uPSI_ShLwApi;                           //Brakel

```

```

20640: 160 unit uPSI_IBUtils;                                //VCL
20641: 161 unit uPSI_CheckLst;                             //VCL
20642: 162 unit uPSI_JvSimpleXml;                           //JCL
20643: 163 unit uPSI_JclSimpleXml;                          //JCL
20644: 164 unit uPSI_JvXmlDatabase;                         //JCL
20645: 165 unit uPSI_JvMaxPixel;                            //JCL
20646: 166 unit uPSI_JvItemsSearchs;                         //JCL
20647: 167 unit uPSI_StExpEng2;                            //SysTools4
20648: 168 unit uPSI_StGenLog;                             //SysTools4
20649: 169 unit uPSI_JvLogFile;                            //Jcl
20650: 170 unit uPSI_CPort;                               //ComPort Lib v4.11
20651: 171 unit uPSI_CPortCtl;                            //ComPort
20652: 172 unit uPSI_CPortEsc;                            //ComPort
20653: 173 unit BarCodeScanner;                           //ComPort
20654: 174 unit uPSI_JvGraph;                            //JCL
20655: 175 unit uPSI_JvComCtrls;                           //JCL
20656: 176 unit uPSI_GUITesting;                          //D Unit
20657: 177 unit uPSI_JvFindFiles;                          //JCL
20658: 178 unit uPSI_StSystem;                            //SysTools4
20659: 179 unit uPSI_JvKeyboardStates;                     //JCL
20660: 180 unit uPSI_JvMail;                             //JCL
20661: 181 unit uPSI_JclConsole;                          //JCL
20662: 182 unit uPSI_JclLANMan;                           //JCL
20663: 183 unit uPSI_IdCustomHTTPServer;                  //Indy
20664: 184 unit IdHTTPServer;                            //Indy
20665: 185 unit uPSI_IdTCPServer;                          //Indy
20666: 186 unit uPSI_IdSocketHandle;                      //Indy
20667: 187 unit uPSI_IdIOHandlerSocket;                   //Indy
20668: 188 unit IdIOHandler;                            //Indy
20669: 189 unit uPSI_cutils;                            //Bloodshed
20670: 190 unit uPSI_BoldUtils;                           //boldsoft
20671: 191 unit uPSI_IdSimpleServer;                      //Indy
20672: 192 unit uPSI_IdSSLOpenSSL;                        //Indy
20673: 193 unit uPSI_IdMultipartFormData;                 //Indy
20674: 194 unit uPSI_SynURIOpener;                        //SynEdit
20675: 195 unit uPSI_PerlRegEx;                           //PCRE
20676: 196 unit uPSI_IdHeaderList;                         //Indy
20677: 197 unit uPSI_StFirst;                            //SysTools4
20678: 198 unit uPSI_JvCtrls;                            //JCL
20679: 199 unit uPSI_IdTrivialFTPBase;                   //Indy
20680: 200 unit uPSI_IdTrivialFTP;                         //Indy
20681: 201 unit uPSI_IdUDPBase;                           //Indy
20682: 202 unit uPSI_IdUDPClient;                          //Indy
20683: 203 unit uPSI_utypes;                            //for DMath.DLL
20684: 204 unit uPSI_ShellAPI;                           //Borland
20685: 205 unit uPSI_IdRemoteCMDClient;                  //Indy
20686: 206 unit uPSI_IdRemoteCMDServer;                  //Indy
20687: 207 unit IdRexecServer;                           //Indy
20688: 208 unit IdRexec; (unit uPSI_IdRexec;)           //Indy
20689: 209 unit IdUDPServer;                            //Indy
20690: 210 unit IdTimeUDPServer;                          //Indy
20691: 211 unit IdTimeServer;                           //Indy
20692: 212 unit IdTimeUDP; (unit uPSI_IdUDPServer;)     //Indy
20693: 213 unit uPSI_IdIPWatch;                           //Indy
20694: 214 unit uPSI_IdIrcServer;                          //Indy
20695: 215 unit uPSI_IdMessageCollection;                //Indy
20696: 216 unit uPSI_cPEM;                             //Fundamentals 4
20697: 217 unit uPSI_cFundamentUtils;                    //Fundamentals 4
20698: 218 unit uPSI_uwinplot;                           //DMath
20699: 219 unit uPSI_xrtl_util_CPUUtils;                 //ExtendedRTL
20700: 220 unit uPSI_GR32_System;                         //Graphics32
20701: 221 unit uPSI_cFileutils;                          //Fundamentals 4
20702: 222 unit uPSI_cDateTime; (timemachine)            //Fundamentals 4
20703: 223 unit uPSI_cTimers; (high precision timer)    //Fundamentals 4
20704: 224 unit uPSI_cRandom;                            //Fundamentals 4
20705: 225 unit uPSI_ueval;                            //DMath
20706: 226 unit uPSI_xrtl_net_URIUtils;                  //ExtendedRTL
20707: 227 unit xrtl_net_URIUtils;                      //ExtendedRTL
20708: 228 unit uPSI_ufft; (FFT)                         //DMath
20709: 229 unit uPSI_DBXChannel;                          //Delphi
20710: 230 unit uPSI_DBXIndyChannel;                     //Delphi Indy
20711: 231 unit uPSI_xrtl_util_COMCat;                  //ExtendedRTL
20712: 232 unit uPSI_xrtl_util_StrUtils;                //ExtendedRTL
20713: 233 unit uPSI_xrtl_util_VariantUtils;             //ExtendedRTL
20714: 234 unit uPSI_xrtl_util_FileUtils;                //ExtendedRTL
20715: 235 unit xrtl_util_Compat;                        //ExtendedRTL
20716: 236 unit uPSI_OleAuto;                            //Borland
20717: 237 unit uPSI_xrtl_util_COMUtils;                 //ExtendedRTL
20718: 238 unit uPSI_CmAdmCtl;                           //Borland
20719: 239 unit uPSI_ValEdit2;                            //VCL
20720: 240 unit uPSI_GR32; //Graphics32                //Graphics32
20721: 241 unit uPSI_GR32_Image;                          //Graphics32
20722: 242 unit uPSI_xrtl_util_TimeUtils;                //ExtendedRTL
20723: 243 unit uPSI_xrtl_util_TimeZone;                 //ExtendedRTL
20724: 244 unit uPSI_xrtl_util_TimeStamp;                //ExtendedRTL
20725: 245 unit uPSI_xrtl_util_Map;                      //ExtendedRTL
20726: 246 unit uPSI_xrtl_util_Set;                      //ExtendedRTL
20727: 247 unit uPSI_CPortMonitor;                         //ComPort
20728: 248 unit uPSI_StIniStm;                            //SysTools4

```

```

20729: 249 unit uPSI_GR32_ExtImage; //Graphics32
20730: 250 unit uPSI_GR32_OrdinalMaps; //Graphics32
20731: 251 unit uPSI_GR32_Rasterizers; //Graphics32
20732: 252 unit uPSI_xrtl_util_Exception; //ExtendedRTL
20733: 253 unit uPSI_xrtl_util_Value; //ExtendedRTL
20734: 254 unit uPSI_xrtl_util_Compare; //ExtendedRTL
20735: 255 unit uPSI_FlatSB; //VCL
20736: 256 unit uPSI_JvAnalogClock; //JCL
20737: 257 unit uPSI_JvAlarms; //JCL
20738: 258 unit uPSI_JvSQLS; //JCL
20739: 259 unit uPSI_JvDBSecur; //JCL
20740: 260 unit uPSI_JvDBQBE; //JCL
20741: 261 unit uPSI_JvStarfield; //JCL
20742: 262 unit uPSI_JvCLMiscal; //JCL
20743: 263 unit uPSI_JvProfiler32; //JCL
20744: 264 unit uPSI_JvDirectories; //JCL
20745: 265 unit uPSI_JclSchedule; //JCL
20746: 266 unit uPSI_JclSvcCtrl; //JCL
20747: 267 unit uPSI_JvSoundControl; //JCL
20748: 268 unit uPSI_JvBDESQLScript; //JCL
20749: 269 unit uPSI_JvgDigits; //JCL>
20750: 270 unit uPSI_ImgList; //TCustomImageList
20751: 271 unit uPSI_JclMIDI; //JCL>
20752: 272 unit uPSI_JclWinMidi; //JCL>
20753: 273 unit uPSI_JclNTFS; //JCL>
20754: 274 unit uPSI_JclAppInst; //JCL>
20755: 275 unit uPSI_JvRle; //JCL>
20756: 276 unit uPSI_JvRas32; //JCL>
20757: 277 unit uPSI_JvImageDrawThread; //JCL>
20758: 278 unit uPSI_JvImageWindow; //JCL>
20759: 279 unit uPSI_JvTransparentForm; //JCL>
20760: 280 unit uPSI_JvWinDialogs; //JCL>
20761: 281 unit uPSI_JvSimLogic; //JCL>
20762: 282 unit uPSI_JvSimIndicator; //JCL>
20763: 283 unit uPSI_JvSimPID; //JCL>
20764: 284 unit uPSI_JvSimPIDLinker; //JCL>
20765: 285 unit uPSI_IdRFCReply; //Indy
20766: 286 unit uPSI_IdIdent; //Indy
20767: 287 unit uPSI_IdIdentServer; //Indy
20768: 288 unit uPSI_JvPatchFile; //JCL
20769: 289 unit uPSI_StNetPfm; //SysTools4
20770: 290 unit uPSI_StNet; //SysTools4
20771: 291 unit uPSI_JclPeImage; //JCL
20772: 292 unit uPSI_JclPrint; //JCL
20773: 293 unit uPSI_JclMime; //JCL
20774: 294 unit uPSI_JvRichEdit; //JCL
20775: 295 unit uPSI_JvDBRichEd; //JCL
20776: 296 unit uPSI_JvDice; //JCL
20777: 297 unit uPSI_JvFloatEdit; //JCL 3.9.8
20778: 298 unit uPSI_JvDirFrm; //JCL
20779: 299 unit uPSI_JvDualList; //JCL
20780: 300 unit uPSI_JvSwitch; //JCL
20781: 301 unit uPSI_JvTimerLst; //JCL
20782: 302 unit uPSI_JvMemTable; //JCL
20783: 303 unit uPSI_JvObjStr; //JCL
20784: 304 unit uPSI_StLArr; //SysTools4
20785: 305 unit uPSI_StWmDCopy; //SysTools4
20786: 306 unit uPSI_StText; //SysTools4
20787: 307 unit uPSI_StNTLog; //SysTools4
20788: 308 unit uPSI_xrtl_math_Integer; //ExtendedRTL
20789: 309 unit uPSI_JvImagPrvw; //JCL
20790: 310 unit uPSI_JvFormPatch; //JCL
20791: 311 unit uPSI_JvPicClip; //JCL
20792: 312 unit uPSI_JvDataConv; //JCL
20793: 313 unit uPSI_JvCpuUsage; //JCL
20794: 314 unit uPSI_JclUnitConv_mx2; //JCL
20795: 315 unit JvDualListForm; //JCL
20796: 316 unit uPSI_JvCpuUsage2; //JCL
20797: 317 unit uPSI_JvParserForm; //JCL
20798: 318 unit uPSI_JvJanTreeView; //JCL
20799: 319 unit uPSI_JvTransLED; //JCL
20800: 320 unit uPSI_JvPlaylist; //JCL
20801: 321 unit uPSI_JvFormAutoSize; //JCL
20802: 322 unit uPSI_JvYearGridEditForm; //JCL
20803: 323 unit uPSI_JvMarkupCommon; //JCL
20804: 324 unit uPSI_JvChart; //JCL
20805: 325 unit uPSI_JvXPCore; //JCL
20806: 326 unit uPSI_JvXPCoreUtils; //JCL
20807: 327 unit uPSI_StatsClasses; //mx4
20808: 328 unit uPSI_ExtCtrls2; //VCL
20809: 329 unit uPSI_JvUrlGrabbers; //JCL
20810: 330 unit uPSI_JvXmlTree; //JCL
20811: 331 unit uPSI_JvWavePlayer; //JCL
20812: 332 unit uPSI_JvUnicodeCanvas; //JCL
20813: 333 unit uPSI_JvTFUtils; //JCL
20814: 334 unit uPSI_IdServerIOHandler; //Indy
20815: 335 unit uPSI_IdServerIOHandlerSocket; //Indy
20816: 336 unit uPSI_IdMessageCoder; //Indy
20817: 337 unit uPSI_IdMessageCoderMIME; //Indy

```

```

20818: 338 unit uPSI_IdMIMETypes; //Indy
20819: 339 unit uPSI_JvConverter; //JCL
20820: 340 unit uPSI_JvCsvParse; //JCL
20821: 341 unit uPSI_umat; unit uPSI_ugamma; //DMath
20822: 342 unit uPSI_ExcelExport; (Nat:TJsExcelExport) //JCL
20823: 343 unit uPSI_JvDBGridExport; //JCL
20824: 344 unit uPSI_JvgExport; //JCL
20825: 345 unit uPSI_JvSerialMaker; //JCL
20826: 346 unit uPSI_JvWin32; //JCL
20827: 347 unit uPSI_JvPaintFX; //JCL
20828: 348 unit uPSI_JvOracleDataSet; (beta) //JCL
20829: 349 unit uPSI_JvValidators; (preview) //JCL
20830: 350 unit uPSI_JvNTEventLog; //JCL
20831: 351 unit uPSI_ShellZipTool; //mx4
20832: 352 unit uPSI_JvJoystick; //JCL
20833: 353 unit uPSI_JvMailSlots; //JCL
20834: 354 unit uPSI_JclComplex; //JCL
20835: 355 unit uPSI_SynPdf; //Synopse
20836: 356 unit uPSI_Registry; //VCL
20837: 357 unit uPSI_TlHelp32; //VCL
20838: 358 unit uPSI_JclRegistry; //JCL
20839: 359 unit uPSI_JvAirBrush; //JCL
20840: 360 unit uPSI_mORMotReport; //Synopse
20841: 361 unit uPSI_JclLocales; //JCL
20842: 362 unit uPSI_SynEdit; //SynEdit
20843: 363 unit uPSI_SynEditTypes; //SynEdit
20844: 364 unit uPSI_SynMacroRecorder; //SynEdit
20845: 365 unit uPSI_LongIntList; //SynEdit
20846: 366 unit uPSI_devutils; //DevC
20847: 367 unit uPSI_SynEditMiscClasses; //SynEdit
20848: 368 unit uPSI_SynEditRegexSearch; //SynEdit
20849: 369 unit uPSI_SynEditHighlighter; //SynEdit
20850: 370 unit uPSI_SynHighlighterPas; //SynEdit
20851: 371 unit uPSI_JvSearchFiles; //JCL
20852: 372 unit uPSI_SynHighlighterAny; //Lazarus
20853: 373 unit uPSI_SynEditKeyCmds; //SynEdit
20854: 374 unit uPSI_SynEditMiscProcs; //SynEdit
20855: 375 unit uPSI_SynEditKbdHandler; //SynEdit
20856: 376 unit uPSI_JvAppInst; //JCL
20857: 377 unit uPSI_JvAppEvent; //JCL
20858: 378 unit uPSI_JvAppCommand; //JCL
20859: 379 unit uPSI_JvAnimTitle; //JCL
20860: 380 unit uPSI_JvAnimatedImage; //JCL
20861: 381 unit uPSI_SynEditExport; //SynEdit
20862: 382 unit uPSI_SynExportHTML; //SynEdit
20863: 383 unit uPSI_SynExportRTF; //SynEdit
20864: 384 unit uPSI_SynEditSearch; //SynEdit
20865: 385 unit uPSI_fMain_back; //maxbox;
20866: 386 unit uPSI_JvZoom; //JCL
20867: 387 unit uPSI_PMrand; //PM
20868: 388 unit uPSI_JvSticker; //JCL
20869: 389 unit uPSI_XmlVerySimple; //mx4
20870: 390 unit uPSI_Services; //ExtPascal
20871: 391 unit uPSI_ExtPascalUtils; //ExtPascal
20872: 392 unit uPSI_SocketsDelphi; //ExtPascal
20873: 393 unit uPSI_StBarC; //SysTools
20874: 394 unit uPSI_StDbBarC; //SysTools
20875: 395 unit uPSI_StBarPN; //SysTools
20876: 396 unit uPSI_StDbPNBC; //SysTools
20877: 397 unit uPSI_StDb2DBC; //SysTools
20878: 398 unit uPSI_StMoney; //SysTools
20879: 399 unit uPSI_JvForth; //JCL
20880: 400 unit uPSI_RestRequest; //mx4
20881: 401 unit uPSI_HttpRESTConnectionIndy; //mx4
20882: 402 unit uPSI_JvXmlDatabase; //update //JCL
20883: 403 unit uPSI_StAstro; //SysTools
20884: 404 unit uPSI_StSort; //SysTools
20885: 405 unit uPSI_StDate; //SysTools
20886: 406 unit uPSI_StDateSt; //SysTools
20887: 407 unit uPSI_StBase; //SysTools
20888: 408 unit uPSI_StVInfo; //SysTools
20889: 409 unit uPSI_JvBrowseFolder; //JCL
20890: 410 unit uPSI_JvBoxProcs; //JCL
20891: 411 unit uPSI_urandom; (unit uranuvag;) //DMath
20892: 412 unit uPSI_usimann; (unit ugenalg;) //DMath
20893: 413 unit uPSI_JvHighlighter; //JCL
20894: 414 unit uPSI_Diff; //mx4
20895: 415 unit uPSI_SpringWinAPI; //DSpring
20896: 416 unit uPSI_StBits; //SysTools
20897: 417 unit uPSI_TomDBQue; //mx4
20898: 418 unit uPSI_MultilangTranslator; //mx4
20899: 419 unit uPSI_HyperLabel; //mx4
20900: 420 unit uPSI_Starter; //mx4
20901: 421 unit uPSI_FileAssocs; //devC
20902: 422 unit uPSI_devFileMonitorX; //devC
20903: 423 unit uPSI_devrund; //devC
20904: 424 unit uPSI_devExec; //devC
20905: 425 unit uPSI_oyxUtils; //devC
20906: 426 unit uPSI_DosCommand; //devC

```

```

20907: 427 unit uPSI_CppTokenizer;                                //devC
20908: 428 unit uPSI_JvHLParser;                                //devC
20909: 429 unit uPSI_JclMap;                                   //JCL
20910: 430 unit uPSI_JclShell;                                 //JCL
20911: 431 unit uPSI_JclCOM;                                  //JCL
20912: 432 unit uPSI_GR32_Math;                                //Graphics32
20913: 433 unit uPSI_GR32_LowLevel;                            //Graphics32
20914: 434 unit uPSI_SimpleHl;                                 //mX4
20915: 435 unit uPSI_GR32_Filters;                             //Graphics32
20916: 436 unit uPSI_GR32_VectorMaps;                          //Graphics32
20917: 437 unit uPSI_cXMLFunctions;                           //Fundamentals 4
20918: 438 unit uPSI_JvTTimer;                                //JCL
20919: 439 unit uPSI_cHTTPUtils;                             //Fundamentals 4
20920: 440 unit uPSI_cTLSUtils;                               //Fundamentals 4
20921: 441 unit uPSI_JclGraphics;                            //JCL
20922: 442 unit uPSI_JclSynch;                               //JCL
20923: 443 unit uPSI_IdTelnet;                               //Indy
20924: 444 unit uPSI_IdTelnetServer;                          //Indy
20925: 445 unit uPSI_IdEcho;                                //Indy
20926: 446 unit uPSI_IdEchoServer;                           //Indy
20927: 447 unit uPSI_IdEchoUDP;                            //Indy
20928: 448 unit uPSI_IdEchoUDPServer;                         //Indy
20929: 449 unit uPSI_IdSocks;                               //Indy
20930: 450 unit uPSI_IdAntiFreezeBase;                        //Indy
20931: 451 unit uPSI_IdHostnameServer;                        //Indy
20932: 452 unit uPSI_IdTunnelCommon;                          //Indy
20933: 453 unit uPSI_IdTunnelMaster;                          //Indy
20934: 454 unit uPSI_IdTunnelSlave;                           //Indy
20935: 455 unit uPSI_IdRSH;                                //Indy
20936: 456 unit uPSI_IdRSHServer;                            //Indy
20937: 457 unit uPSI_Spring_Cryptography_Utils;             //Spring4Delphi
20938: 458 unit uPSI_MapReader;                             //devC
20939: 459 unit uPSI_LibTar;                                //devC
20940: 460 unit uPSI_IdStack;                               //Indy
20941: 461 unit uPSI_IdBlockCipherIntercept;                //Indy
20942: 462 unit uPSI_IdChargenServer;                        //Indy
20943: 463 unit uPSI_IdFTPServer;                            //Indy
20944: 464 unit uPSI_IdException;                           //Indy
20945: 465 unit uPSI_utexplot;                             //DMath
20946: 466 unit uPSI_uwinstr;                             //DMath
20947: 467 unit uPSI_VarRecUtils;                           //devC
20948: 468 unit uPSI_JvStringListToHtml;                     //JCL
20949: 469 unit uPSI_JvStringHolder;                          //JCL
20950: 470 unit uPSI_IdCoder;                               //Indy
20951: 471 unit uPSI_SynHighlighterDfm;                      //Synedit
20952: 472 unit uHighlighterProcs; in 471                  //Synedit
20953: 473 unit uPSI_LazFileUtils;                           //LCL
20954: 474 unit uPSI_IDECmdLine;                            //LCL
20955: 475 unit uPSI_lazMasks;                             //LCL
20956: 476 unit uPSI_ip_misc;                             //mX4
20957: 477 unit uPSI_Barcodes;                            //LCL
20958: 478 unit uPSI_SimpleXML;                            //LCL
20959: 479 unit uPSI_JclIniFiles;                           //JCL
20960: 480 unit uPSI_D2XXUnit; { $X- }                   //FTDI
20961: 481 unit uPSI_JclDateTime;                           //JCL
20962: 482 unit uPSI_JclEDI;                              //JCL
20963: 483 unit uPSI_JclMiscel2;                           //JCL
20964: 484 unit uPSI_JclValidation;                         //JCL
20965: 485 unit uPSI_JclAnsiStrings; {-PString}          //JCL
20966: 486 unit uPSI_SynEditMiscProcs2;                    //Synedit
20967: 487 unit uPSI_JclStreams;                           //JCL
20968: 488 unit uPSI_QRCode;                             //mX4
20969: 489 unit uPSI_BlockSocket;                          //ExtPascal
20970: 490 unit uPSI_Masks_Utils;                           //VCL
20971: 491 unit uPSI_synautil;                            //Synapse!
20972: 492 unit uPSI_JclMath_Class;                         //JCL RTL
20973: 493 unit ugamdist; //Gamma function               //DMath
20974: 494 unit uibeta, ucorrel; //IBeta                 //DMath
20975: 495 unit uPSI_SRMgr;                               //mX4
20976: 496 unit uPSI_HotLog;                             //mX4
20977: 497 unit uPSI_DebugBox;                            //mX4
20978: 498 unit uPSI_ustrings;                           //DMath
20979: 499 unit uPSI_uregtest;                            //DMath
20980: 500 unit uPSI_usimplex;                           //DMath
20981: 501 unit uPSI_uhyper;                             //DMath
20982: 502 unit uPSI_IdHL7;                             //Indy
20983: 503 unit uPSI_IdIPMCastBase;                      //Indy
20984: 504 unit uPSI_IdIPMCastServer;                    //Indy
20985: 505 unit uPSI_IdIPMCastClient;                   //Indy
20986: 506 unit uPSI_unlfit; //nlregression           //DMath
20987: 507 unit uPSI_IdRawHeaders;                        //Indy
20988: 508 unit uPSI_IdRawClient;                          //Indy
20989: 509 unit uPSI_IdRawFunctions;                      //Indy
20990: 510 unit uPSI_IdTCPStream;                          //Indy
20991: 511 unit uPSI_IdSNPP;                            //Indy
20992: 512 unit uPSI_St2DBarC;                           //SysTools
20993: 513 unit uPSI_ImageWin; //FTL                  //VCL
20994: 514 unit uPSI_CustomDrawTreeView; //FTL            //VCL
20995: 515 unit uPSI_GraphWin; //FTL                  //VCL

```

```

20996: 516 unit uPSI_actionMain; //FTL          //VCL
20997: 517 unit uPSI_StSpawn;                  //SysTools
20998: 518 unit uPSI_CtlPanel;                 //VCL
20999: 519 unit uPSI_IdLPR;                   //Indy
21000: 520 unit uPSI_SockRequestInterpreter;   //Indy
21001: 521 unit uPSI_umlambert;                //DMath
21002: 522 unit uPSI_ucholesk;                 //DMath
21003: 523 unit uPSI_SimpleDS;                 //VCL
21004: 524 unit uPSI_DBXSqlScanner;            //VCL
21005: 525 unit uPSI_DBXMetaDataTable;         //VCL
21006: 526 unit uPSI_Chart;                   //TEE
21007: 527 unit uPSI_TeeProcs;                 //TEE
21008: 528 unit mXBDEUtils;                   //mX4
21009: 529 unit uPSI_MDIEdit;                  //VCL
21010: 530 unit uPSI_CopyPsr;                  //VCL
21011: 531 unit uPSI_SockApp;                 //VCL
21012: 532 unit uPSI_AppEvnts;                //VCL
21013: 533 unit uPSI_ExtActns;                //VCL
21014: 534 unit uPSI_TeEngine;                 //TEE
21015: 535 unit uPSI_CoolMain; //browser      //VCL
21016: 536 unit uPSI_StCRC;                   //SysTools
21017: 537 unit uPSI_StDecMth2;               //SysTools
21018: 538 unit uPSI_frmExportMain;            //Synedit
21019: 539 unit uPSI_SynDBEdit;                //Synedit
21020: 540 unit uPSI_SynEditWildcardSearch;    //Synedit
21021: 541 unit uPSI_BoldComUtils;             //BOLD
21022: 542 unit uPSI_BoldIsoDateTime;          //BOLD
21023: 543 unit uPSI_BoldGUIDUtils;            //inCOMUtils
21024: 544 unit uPSI_BoldXMLRequests;          //BOLD
21025: 545 unit uPSI_BoldStringList;            //BOLD
21026: 546 unit uPSI_BoldfileHandler;          //BOLD
21027: 547 unit uPSI_BoldContainers;           //BOLD
21028: 548 unit uPSI_BoldQueryUserDlg;          //BOLD
21029: 549 unit uPSI_BoldWinINet;               //BOLD
21030: 550 unit uPSI_BoldQueue;                 //BOLD
21031: 551 unit uPSI_JvPcx;                   //JCL
21032: 552 unit uPSI_IdWhois;                  //Indy
21033: 553 unit uPSI_IdWhoisServer;            //Indy
21034: 554 unit uPSI_IdGopher;                 //Indy
21035: 555 unit uPSI_IdDateTimeStamp;          //Indy
21036: 556 unit uPSI_IdDayTimeServer;          //Indy
21037: 557 unit uPSI_IdDayTimeUDP;             //Indy
21038: 558 unit uPSI_IdDayTimeUDPServer;       //Indy
21039: 559 unit uPSI_IdDICTServer;             //Indy
21040: 560 unit uPSI_IdDiscardServer;          //Indy
21041: 561 unit uPSI_IdDiscardUDPServer;       //Indy
21042: 562 unit uPSI_IdMappedFTP;              //Indy
21043: 563 unit uPSI_IdMappedPortTCP;           //Indy
21044: 564 unit uPSI_IdGopherServer;            //Indy
21045: 565 unit uPSI_IdQotdServer;             //Indy
21046: 566 unit uPSI_JvRgbToHtml;               //JCL
21047: 567 unit uPSI_JvRemLog;                 //JCL
21048: 568 unit uPSI_JvSysComp;                //JCL
21049: 569 unit uPSI_JvTMTL;                  //JCL
21050: 570 unit uPSI_JvWinampAPI;               //JCL
21051: 571 unit uPSI_MSysUtils;                //mX4
21052: 572 unit uPSI_ESBMaths;                 //ESB
21053: 573 unit uPSI_ESBMaths2;                //ESB
21054: 574 unit uPSI_uLkJSON;                  //Lk
21055: 575 unit uPSI_ZURL; //Zeos             //Zeos
21056: 576 unit uPSI_ZSysUtils;                //Zeos
21057: 577 unit unaUtils internals;             //UNA
21058: 578 unit uPSI_ZMatchPattern;            //Zeos
21059: 579 unit uPSI_ZClasses;                 //Zeos
21060: 580 unit uPSI_ZCollections;             //Zeos
21061: 581 unit uPSI_ZEncoding;                //Zeos
21062: 582 unit uPSI_IdRawBase;                //Indy
21063: 583 unit uPSI_IdNTLM;                  //Indy
21064: 584 unit uPSI_IdNNTP;                  //Indy
21065: 585 unit uPSI_usniffer; //PortScanForm //mX4
21066: 586 unit uPSI_IdCoderMIME;              //Indy
21067: 587 unit uPSI_IdCoderUUE;               //Indy
21068: 588 unit uPSI_IdCoderXXE;               //Indy
21069: 589 unit uPSI_IdCoder3to4;              //Indy
21070: 590 unit uPSI_IdCookie;                 //Indy
21071: 591 unit uPSI_IdCookieManager;          //Indy
21072: 592 unit uPSI_WdosSocketUtils;          //WDos
21073: 593 unit uPSI_WdosPlcUtils;             //WDos
21074: 594 unit uPSI_WdosPorts;                //WDos
21075: 595 unit uPSI_WdosResolvers;            //WDos
21076: 596 unit uPSI_WdosTimers;               //WDos
21077: 597 unit uPSI_WdosPlcs;                 //WDos
21078: 598 unit uPSI_WdosPneumatics;           //WDos
21079: 599 unit uPSI_IdFingerServer;            //Indy
21080: 600 unit uPSI_IdDNSResolver;             //Indy
21081: 601 unit uPSI_IdHTTPWebBrokerBridge;    //Indy
21082: 602 unit uPSI_IdIntercept;               //Indy
21083: 603 unit uPSI_IdIPMCastBase;             //Indy
21084: 604 unit uPSI_IdLogBase;                 //Indy

```

```

21085: 605 unit uPSI_IdIOHandlerStream;           //Indy
21086: 606 unit uPSI_IdMappedPortUDP;           //Indy
21087: 607 unit uPSI_IdQOTDUDPServer;          //Indy
21088: 608 unit uPSI_IdQOTDUDP;                //Indy
21089: 609 unit uPSI_IdSysLog;                 //Indy
21090: 610 unit uPSI_IdSysLogServer;            //Indy
21091: 611 unit uPSI_IdSysLogMessage;           //Indy
21092: 612 unit uPSI_IdTimeServer;              //Indy
21093: 613 unit uPSI_IdTimeUDP;                //Indy
21094: 614 unit uPSI_IdTimeUDPServer;           //Indy
21095: 615 unit uPSI_IdUserAccounts;            //Indy
21096: 616 unit uPSI_TextUtils;                 //mX4
21097: 617 unit uPSI_MandelbrotEngine;          //mX4
21098: 618 unit uPSI_delphi_arduino_Unit1;       //mX4
21099: 619 unit uPSI_DTDSchema2;                //mX4
21100: 620 unit uPSI_fpplotMain;                //DMath
21101: 621 unit uPSI_FindFileIter;               //mX4
21102: 622 unit uPSI_PppState; (JclStrHashMap) //PPP
21103: 623 unit uPSI_PppParser;                 //PPP
21104: 624 unit uPSI_PppLexer;                 //PPP
21105: 625 unit uPSI_PCharUtils;                //PPP
21106: 626 unit uPSI_uJSON;                     //WU
21107: 627 unit uPSI_JclStrHashMap;             //JCL
21108: 628 unit uPSI_JclHookExcept;             //JCL
21109: 629 unit uPSI_EncdDecd;                 //VCL
21110: 630 unit uPSI_SockAppReg;                //VCL
21111: 631 unit uPSI_PJFileHandle;              //PJ
21112: 632 unit uPSI_PJEnvVars;                //PJ
21113: 633 unit uPSI_PJPipe;                   //PJ
21114: 634 unit uPSI_PJPipeFilters;             //PJ
21115: 635 unit uPSI_PJConsoleApp;              //PJ
21116: 636 unit uPSI_UConsoleAppEx;             //PJ
21117: 637 unit uPSI_DbxBSocketChannelNative;   //VCL
21118: 638 unit uPSI_DbxDDataGenerator;          //VCL
21119: 639 unit uPSI_DBXClient;                 //VCL
21120: 640 unit uPSI_IdLogEvent;                //Indy
21121: 641 unit uPSI_Reversi;                  //mX4
21122: 642 unit uPSI_Geometry;                 //mX4
21123: 643 unit uPSI_IdSMTPServer;              //Indy
21124: 644 unit uPSI_Textures;                 //mX4
21125: 645 unit uPSI_IBX;                      //VCL
21126: 646 unit uPSI_IWDBCommon;                //VCL
21127: 647 unit uPSI_SortGrid;                 //mX4
21128: 648 unit uPSI_IB;                       //VCL
21129: 649 unit uPSI_IBScript;                 //VCL
21130: 650 unit uPSI_JvCSVBaseControls;         //JCL
21131: 651 unit uPSI_Jvg3DColors;              //JCL
21132: 652 unit uPSI_JvHLEditor; //beat        //JCL
21133: 653 unit uPSI_JvShellHook;              //JCL
21134: 654 unit uPSI_DBCommon2;                //VCL
21135: 655 unit uPSI_JvSHfileOperation;         //JCL
21136: 656 unit uPSI_uFilelexport;             //mX4
21137: 657 unit uPSI_JvDialogs;                //JCL
21138: 658 unit uPSI_JvDBTreeView;              //JCL
21139: 659 unit uPSI_JvDBUltimGrid;             //JCL
21140: 660 unit uPSI_JvDBQueryParamsForm;       //JCL
21141: 661 unit uPSI_JvExControls;              //JCL
21142: 662 unit uPSI_JvBDEMemTable;             //JCL
21143: 663 unit uPSI_JvCommStatus;              //JCL
21144: 664 unit uPSI_JvMailSlots2;              //JCL
21145: 665 unit uPSI_JvgWinMask;                //JCL
21146: 666 unit uPSI_StEclipse;                //SysTools
21147: 667 unit uPSI_StMime;                   //SysTools
21148: 668 unit uPSI_StList;                   //SysTools
21149: 669 unit uPSI_StMerge;                  //SysTools
21150: 670 unit uPSI_StStrs;                  //SysTools
21151: 671 unit uPSI_StTree;                   //SysTools
21152: 672 unit uPSI_StVArr;                   //SysTools
21153: 673 unit uPSI_StRegIni;                //SysTools
21154: 674 unit uPSI_urkf;                    //DMath
21155: 675 unit uPSI_usvd;                    //DMath
21156: 676 unit uPSI_DepWalkUtils;             //JCL
21157: 677 unit uPSI_OptionsFrm;              //JCL
21158: 678 unit yuvconverts;                  //mX4
21159: 679 uPSI_JvPropAutoSave;                //JCL
21160: 680 uPSI_AclAPI;                      //alcinoe
21161: 681 uPSI_AviCap;                      //alcinoe
21162: 682 uPSI_ALAVLBinaryTree;              //alcinoe
21163: 683 uPSI_ALFcMisc;                    //alcinoe
21164: 684 uPSI_ALStringList;                //alcinoe
21165: 685 uPSI_ALQuickSortList;              //alcinoe
21166: 686 uPSI_ALStaticText;                //alcinoe
21167: 687 uPSI_ALJSONDoc;                  //alcinoe
21168: 688 uPSI_ALGSMComm;                  //alcinoe
21169: 689 uPSI_ALWindows;                  //alcinoe
21170: 690 uPSI_ALMultiPartFormDataParser;    //alcinoe
21171: 691 uPSI_ALHttpCommon;                //alcinoe
21172: 692 uPSI_ALWebSpider;                //alcinoe
21173: 693 uPSI_ALHttpClient;                //alcinoe

```

```

21174: 694 uPSI_ALFcHTML; //alcinoe
21175: 695 uPSI_ALFTPClient; //alcinoe
21176: 696 uPSI_ALInternetMessageCommon; //alcinoe
21177: 697 uPSI_ALWininetHttpclient; //alcinoe
21178: 698 uPSI_ALWinInetFTPCClient; //alcinoe
21179: 699 uPSI_ALWinHttpWrapper; //alcinoe
21180: 700 uPSI_ALWinHttpClient; //alcinoe
21181: 701 uPSI_ALFcWinSock; //alcinoe
21182: 702 uPSI_ALFcSQL; //alcinoe
21183: 703 uPSI_ALFcCGI; //alcinoe
21184: 704 uPSI_ALFcExecute; //alcinoe
21185: 705 uPSI_ALFcFile; //alcinoe
21186: 706 uPSI_ALFcMime; //alcinoe
21187: 707 uPSI_ALPhpRunner; //alcinoe
21188: 708 uPSI_ALGraphic; //alcinoe
21189: 709 uPSI_ALIniFiles; //alcinoe
21190: 710 uPSI_ALMemCachedClient; //alcinoe
21191: 711 unit uPSI_MyGrids; //mx4
21192: 712 uPSI_ALMultiPartMixedParser //alcinoe
21193: 713 uPSI_ALSMTPClient //alcinoe
21194: 714 uPSI_ALNNTPClient //alcinoe
21195: 715 uPSI_ALHintBalloon; //alcinoe
21196: 716 unit uPSI_ALXmlDoc; //alcinoe
21197: 717 unit uPSI_IPCThrd; //VCL
21198: 718 unit uPSI_MonForm; //VCL
21199: 719 unit uPSI_TeCanvas; //Orpheus
21200: 720 unit uPSI_OvcMisc; //Orpheus
21201: 721 unit uPSI_ovcfiler; //Orpheus
21202: 722 unit uPSI_ovcstate; //Orpheus
21203: 723 unit uPSI_ovccoco; //Orpheus
21204: 724 unit uPSI_ovcrvexp; //Orpheus
21205: 725 unit uPSI_OvcFormatSettings; //Orpheus
21206: 726 unit uPSI_OvcUtils; //Orpheus
21207: 727 unit uPSI_ovcstore; //Orpheus
21208: 728 unit uPSI_ovcstr; //Orpheus
21209: 729 unit uPSI_ovcmru; //Orpheus
21210: 730 unit uPSI_ovccmd; //Orpheus
21211: 731 unit uPSI_ovctimer; //Orpheus
21212: 732 unit uPSI_ovcintl; //Orpheus
21213: 733 uPSI_AfCircularBuffer; //AsyncFree
21214: 734 uPSI_AfUtils; //AsyncFree
21215: 735 uPSI_AfSafeSync; //AsyncFree
21216: 736 uPSI_AfComPortCore; //AsyncFree
21217: 737 uPSI_AfComPort; //AsyncFree
21218: 738 uPSI_AfPortControls; //AsyncFree
21219: 739 uPSI_AfDataDispatcher; //AsyncFree
21220: 740 uPSI_AfViewers; //AsyncFree
21221: 741 uPSI_AfDataTerminal; //AsyncFree
21222: 742 uPSI_SimplePortMain; //AsyncFree
21223: 743 unit uPSI_ovcclock; //Orpheus
21224: 744 unit uPSI_o32intlst; //Orpheus
21225: 745 unit uPSI_o32ledlabel; //Orpheus
21226: 746 unit uPSI_AlMySqlClient; //alcinoe
21227: 747 unit uPSI_ALFBXClient; //alcinoe
21228: 748 unit uPSI_ALFcSQL; //alcinoe
21229: 749 unit uPSI_AsyncTimer; //mx4
21230: 750 unit uPSI_ApplicationFileIO; //mx4
21231: 751 unit uPSI_PsAPI; //VCLé
21232: 752 uPSI_ovcuser; //Orpheus
21233: 753 uPSI_ovcurl; //Orpheus
21234: 754 uPSI_ovcvlb; //Orpheus
21235: 755 uPSI_ovccolor; //Orpheus
21236: 756 uPSI_ALFBXLib; //alcinoe
21237: 757 uPSI_ovcmeter; //Orpheus
21238: 758 uPSI_ovcpeakm; //Orpheus
21239: 759 uPSI_O32BGsty; //Orpheus
21240: 760 uPSI_ovcBidi; //Orpheus
21241: 761 uPSI_ovctcarry; //Orpheus
21242: 762 uPSI_DXPUtils; //mx4
21243: 763 uPSI_ALMultiPartBaseParser; //alcinoe
21244: 764 uPSI_ALMultiPartAlternativeParser; //alcinoe
21245: 765 uPSI_ALPOP3Client; //alcinoe
21246: 766 uPSI_SmallUtils; //mx4
21247: 767 uPSI_MakeApp; //mx4
21248: 768 uPSI_O32MouseMon; //Orpheus
21249: 769 uPSI_OvcCache; //Orpheus
21250: 770 uPSI_ovccalc; //Orpheus
21251: 771 uPSI_Joystick; //OpenGL
21252: 772 uPSI_ScreenSaver; //OpenGL
21253: 773 uPSI_XCollection; //OpenGL
21254: 774 uPSI_Polynomials; //OpenGL
21255: 775 uPSI_PersistentClasses, //9.86 //OpenGL
21256: 776 uPSI_VectorLists; //OpenGL
21257: 777 uPSI_XOpenGL; //OpenGL
21258: 778 uPSI_MeshUtils; //OpenGL
21259: 779 unit uPSI_JclSysUtils; //JCL
21260: 780 unit uPSI_JclBorlandTools; //JCL
21261: 781 unit JclFileUtils_max; //JCL
21262: 782 uPSI_AfDataControls, //AsyncFree

```

```

21263: 783 uPSI_GLSilhouette; //OpenGL
21264: 784 uPSI_JclSysUtils_class; //JCL
21265: 785 uPSI_JclFileUtils_class; //JCL
21266: 786 uPSI_FileUtil; //mX4
21267: 787 uPSI_changefind; //mX4
21268: 788 uPSI_CmdIntf; //mX4
21269: 789 uPSI_fservice; //mX4
21270: 790 uPSI_Keyboard; //OpenGL
21271: 791 uPSI_VRMLParser; //OpenGL
21272: 792 uPSI_GLFileVRML; //OpenGL
21273: 793 uPSI_Octree; //OpenGL
21274: 794 uPSI_GLPolyhedron; //OpenGL
21275: 795 uPSI_GLCrossPlatform; //OpenGL
21276: 796 uPSI_GLParticles; //OpenGL
21277: 797 uPSI_GLNavigator; //OpenGL
21278: 798 uPSI_GLStarRecord; //OpenGL
21279: 799 uPSI_GLTextureCombiners; //OpenGL
21280: 800 uPSI_GLCanvas; //OpenGL
21281: 801 uPSI_GeometryBB; //OpenGL
21282: 802 uPSI_GeometryCoordinates; //OpenGL
21283: 803 uPSI_VectorGeometry; //OpenGL
21284: 804 uPSI_BumpMapping; //OpenGL
21285: 805 uPSI_TGA; //OpenGL
21286: 806 uPSI_GLVectorFileObjects; //OpenGL
21287: 807 uPSI_IMM; //VCL
21288: 808 uPSI_CategoryButtons; //VCL
21289: 809 uPSI_ButtonGroup; //VCL
21290: 810 uPSI_DbExcept; //VCL
21291: 811 uPSI_AxCtrls; //VCL
21292: 812 uPSI_GL_actorUnitl; //OpenGL
21293: 813 uPSI_StdVCL; //VCL
21294: 814 unit CurvesAndSurfaces; //OpenGL
21295: 815 uPSI_DataAwareMain; //AsyncFree
21296: 816 uPSI_TabNotBk; //VCL
21297: 817 uPSI_udwsfiler; //mX4
21298: 818 uPSI_synaip; //Synapse!
21299: 819 uPSI_synacode; //Synapse
21300: 820 uPSI_synachar; //Synapse
21301: 821 uPSI_synamisc; //Synapse
21302: 822 uPSI_synaser; //Synapse
21303: 823 uPSI_synaicnv; //Synapse
21304: 824 uPSI_tlntrysend; //Synapse
21305: 825 uPSI_pingsend; //Synapse
21306: 826 uPSI_blksock; //Synapse
21307: 827 uPSI_asnlutil; //Synapse
21308: 828 uPSI_dnssend; //Synapse
21309: 829 uPSI_clamsend; //Synapse
21310: 830 uPSI_ldapsend; //Synapse
21311: 831 uPSI_mimemess; //Synapse
21312: 832 uPSI_slogsend; //Synapse
21313: 833 uPSI_mimepart; //Synapse
21314: 834 uPSI_mimeinln; //Synapse
21315: 835 uPSI_ftpsend; //Synapse
21316: 836 uPSI_ftptsend; //Synapse
21317: 837 uPSI_httpsend; //Synapse
21318: 838 uPSI_ntpssend; //Synapse
21319: 839 uPSI_smtpsend; //Synapse
21320: 840 uPSI_snmpsend; //Synapse
21321: 841 uPSI_imapsend; //Synapse
21322: 842 uPSI_pop3send; //Synapse
21323: 843 uPSI_nntpsend; //Synapse
21324: 844 uPSI_ssl_cryptlib; //Synapse
21325: 845 uPSI_ssl_openssl; //Synapse
21326: 846 uPSI_synhttp_daemon; //Synapse
21327: 847 uPSI_NetWork; //mX4
21328: 848 uPSI_PingThread; //Synapse
21329: 849 uPSI_JvThreadTimer; //JCL
21330: 850 unit uPSI_wwSystem; //InfoPower
21331: 851 unit uPSI_IdComponent; //Indy
21332: 852 unit uPSI_IdIOHandlerThrottle; //Indy
21333: 853 unit uPSI_Themes; //VCL
21334: 854 unit uPSI_StdStyleActnCtrls; //VCL
21335: 855 unit uPSI_UDDIHelper; //VCL
21336: 856 unit uPSI_IdIMAP4Server; //Indy
21337: 857 uPSI_VariantSymbolTable; //VCL //3.9.9.92
21338: 858 uPSI_udf_glob; //mX4
21339: 859 uPSI_TabGrid; //VCL
21340: 860 uPSI_JsDBTreeView; //mX4
21341: 861 uPSI_JsSendMail; //mX4
21342: 862 uPSI_dbTvRecordList; //mX4
21343: 863 uPSI_TreeWEx; //mX4
21344: 864 uPSI_ECDataLink; //mX4
21345: 865 uPSI_dbTree; //mX4
21346: 866 uPSI_dbTreeCBox; //mX4
21347: 867 unit uPSI_Debug; //TfrmDebug
21348: 868 uPSI_TimeFncts; //mX4
21349: 869 uPSI_FileIntf; //VCL
21350: 870 uPSI_SockTransport; //RTL
21351: 871 unit uPSI_WinInet; //RTL

```

```

21352: 872 unit uPSI_Wwstr; //mX4
21353: 873 uPSI_DBLookup; //VCL
21354: 874 uPSI_Hotspot; //mX4
21355: 875 uPSI_HList; //History List //mX4
21356: 876 unit uPSI_DrTable; //VCL
21357: 877 uPSI_TConnect; //VCL
21358: 978 uPSI_DataBkr; //VCL
21359: 979 uPSI_HTTPIntr; //VCL
21360: 980 unit uPSI_Mathbox; //mX4
21361: 881 uPSI_cyIndy; //cY
21362: 882 uPSI_cySysUtils; //cY
21363: 883 uPSI_cyWinUtils; //cY
21364: 884 uPSI_cyStrUtils; //cY
21365: 885 uPSI_cyObjUtils; //cY
21366: 886 uPSI_cyDateUtils; //cY
21367: 887 uPSI_cyBDE; //cY
21368: 888 uPSI_cyClasses; //cY
21369: 889 uPSI_cyGraphics; //3.9.9.94_2 //cY
21370: 890 unit uPSI_cyTypes; //cY
21371: 891 uPSI_JvDateTimePicker; //JCL
21372: 892 uPSI_JvCreateProcess; //JCL
21373: 893 uPSI_JvEasterEgg; //JCL
21374: 894 uPSI_WinSvc; //3.9.9.94_3 //VCL
21375: 895 uPSI_SvcMgr //VCL
21376: 896 unit uPSI_JvPickDate; //JCL
21377: 897 unit uPSI_JvNotify; //JCL
21378: 898 uPSI_JvStrHlder //JCL
21379: 899 unit uPSI_JclNTFS2; //JCL
21380: 900 uPSI_Jcl8087 //math coprocessor //JCL
21381: 901 uPSI_JvAddPrinter //JCL
21382: 902 uPSI_JvCabFile //JCL
21383: 903 uPSI_JvDataEmbedded; //JCL
21384: 904 unit uPSI_U_HexView; //mX4
21385: 905 uPSI_UWavein4; //mX4
21386: 906 uPSI_AMixer; //mX4
21387: 907 uPSI_JvaScrollText; //mX4
21388: 908 uPSI_JvArrow; //mX4
21389: 909 unit uPSI.UrlMon; //mX4
21390: 910 U_Oscilloscope4 in 'U_Oscilloscope4.pas' //mX4
21391: 911 unit uPSI_U_Oscilloscope4; //ToscfirMain; //DFF
21392: 912 unit uPSI_DFFUtils; //DFF
21393: 913 unit uPSI_MathsLib; //DFF
21394: 914 uPSI_UIntlist; //DFF
21395: 915 uPSI_UGetParens; //DFF
21396: 916 unit uPSI_UGeometry; //DFF
21397: 917 unit uPSI_UAstronomy; //DFF
21398: 918 unit uPSI_UCardComponentV2; //DFF
21399: 919 unit uPSI_UTGraphSearch; //DFF
21400: 920 unit uPSI_UParser10; //DFF
21401: 921 unit uPSI_cyIEUtils; //cY
21402: 922 unit uPSI_UcomboV2; //DFF
21403: 923 uPSI_cyBaseComm; //cY
21404: 924 uPSI_cyAppInstances; //cY
21405: 925 uPSI_cyAttract; //cY
21406: 926 uPSI_cyDERUtils; //cY
21407: 927 unit uPSI_cyDocER; //cY
21408: 928 unit uPSI_ODBC; //mX
21409: 929 unit uPSI_AssocExec; //mX
21410: 930 uPSI_cyBaseCommRoomConnector; //cY
21411: 931 uPSI_cyCommRoomConnector; //cY
21412: 932 uPSI_cyCommunicate; //cY
21413: 933 uPSI_cyImage; //cY
21414: 934 uPSI_cyBaseContainer; //cY
21415: 935 uPSI_cyModalContainer; //cY
21416: 936 uPSI_cyFlyingContainer; //cY
21417: 937 uPSI_RegStr; //VCL
21418: 938 uPSI_HtmlHelpViewer; //VCL
21419: 939 unit uPSI_cyIniform; //cY
21420: 940 unit uPSI_cyVirtualGrid; //cY
21421: 941 uPSI_Profiler; //DA
21422: 942 uPSI_BackgroundWorker; //DA
21423: 943 uPSI_Waveplay; //DA
21424: 944 uPSI_WaveTimer; //DA
21425: 945 uPSI_WaveUtils; //DA
21426: 946 uPSI_NamedPipes; //TB
21427: 947 uPSI_NamedPipeServer; //TB
21428: 948 unit uPSI_process; //TB
21429: 949 unit uPSI_DPUtis; //TB
21430: 950 unit uPSI_CommonTools; //TB
21431: 951 uPSI_DataSendToWeb; //TB
21432: 952 uPSI_StarCalc; //TB
21433: 953 uPSI_D2_XPVistaHelperU //TB
21434: 954 unit uPSI_NetTools //TB
21435: 955 unit uPSI_Pipes //TB
21436: 956 uPSI_ProcessUnit; //mX
21437: 957 uPSI_adGSM; //mX
21438: 958 unit uPSI_BetterADODataSet; //mX
21439: 959 unit uPSI_AdSelCom; //FTT //mX
21440: 960 unit unit uPSI_dwsXplatform; //DWS

```

```

21441: 961 uPSI_AdSocket; //mX Turbo Power
21442: 962 uPSI_AdPacket; //mX
21443: 963 uPSI_AdPort; //mX
21444: 964 uPSI_PathFunc; //Inno
21445: 965 uPSI_CmnFunc; //Inno
21446: 966 uPSI_CmnFunc2; //Inno Setup
21447:
21448:
21449: /////////////////////////////////
21450: //Form Template Library FTL
21451: /////////////////////////////////
21452:
21453: 32 FTL For Form Building out of the Script, eg. 399_form_templates.txt
21454:
21455: 045 unit uPSI_VListView TFormListView;
21456: 263 unit uPSI_JvProfiler32; TProfReport
21457: 270 unit uPSI_ImgList; TCustomImageList
21458: 278 unit uPSI_JvImageWindow; TJvImageWindow
21459: 317 unit uPSI_JvParserForm; TJvHTMLParserForm
21460: 497 unit uPSI_DebugBox; TDebugBox
21461: 513 unit uPSI_ImageWin; TImageForm, TImageForm2
21462: 514 unit uPSI_CustomDrawTreeView; TCustomDrawForm
21463: 515 unit uPSI_GraphWin; TGraphWinForm
21464: 516 unit uPSI_actionMain; TActionForm
21465: 518 unit uPSI_CtlPanel; TAppletApplication
21466: 529 unit uPSI_MDIEdit; TEditForm
21467: 535 unit uPSI_CoolMain; {browser} TWebMainForm
21468: 538 unit uPSI_frmExportMain; TSynexportForm
21469: 585 unit uPSI_usniffer; //PortScanForm
21470: 600 unit uPSI_ThreadForm; TThreadSortForm
21471: 618 unit uPSI_delphi_arduino_Unit1; TLEDForm
21472: 620 unit uPSI_fplotMain; TfplotForm1
21473: 660 unit uPSI_JvDBQueryParamsForm; TJvQueryParamsDialog
21474: 677 unit uPSI_OptionsFrm; TfrmOptions;
21475: 718 unit uPSI_MonForm; TMonitorForm
21476: 742 unit uPSI_SimplePortMain; TPortForm1
21477: 770 unit uPSI_occcalc; TOvcCalculator //widget
21478: 810 unit uPSI_DbExcept; TDbEngineErrorDlg
21479: 812 unit uPSI_GL_actorUnit1; TglActorForm1 //OpenGL Robot
21480: 846 unit uPSI_syntht_daemon; TTCPHttpDaemon, TTCPHttpThrd, TPingThread
21481: 867 unit uPSI_Debug; TfrmDebug
21482: 904 unit uPSI_U_HexView; THexForm2
21483: 911 unit uPSI_U_Oscilloscope4; (OscfrmMain)
21484: 959 unit uPSI_AdSelCom; TOscfrmMain
21485:
21486:
21487: ex.:with TEditForm.create(self) do begin
21488:   caption:= 'Template Form Tester';
21489:   FormStyle:= fsStayOnTop;
21490:   with editor do begin
21491:     Lines.LoadFromFile(Exepath+'\docs\Readme_rus_mX2.rtf'
21492:     SelStart:= 0;
21493:     Modified:= False;
21494:   end;
21495: end;
21496: with TWebMainForm.create(self) do begin
21497:   URLs.Text:= 'http://www.kleiner.ch';
21498:   URLsClick(self); Show;
21499: end;
21500: with TSynexportForm.create(self) do begin
21501:   Caption:= 'Synexport HTML RTF tester';
21502:   Show;
21503: end;
21504: with TThreadSortForm.create(self) do begin
21505:   showmodal; free;
21506: end;
21507: with TCustomDrawForm.create(self) do begin
21508:   width:=820; height:=820;
21509:   image1.height:= 600; //add properties
21510:   image1.picture.bitmap:= image2.picture.bitmap;
21511:   //SelectionBackground1Click(self) CustomDraw1Click(self);
21512:   Background1.click;
21513:   bitmpl.click;
21514:   Tile1.click;
21515:   Showmodal;
21516:   Free;
21517: end;
21518: with TfplotForm1.Create(self) do begin
21519:   BtnPlotClick(self);
21520:   Showmodal; Free;
21521: end;
21522: with TOvcCalculator.create(self) do begin
21523:   parent:= aForm;
21524:   //free;
21525:   setbounds(550,510,200,150);
21526:   displaystr:= 'maXcalc';
21527: end;
21528: with THexForm2.Create(self) do begin
21529:   ShowModal;

```

```

21530:     Free;
21531:   end;
21532:
21533: function CheckBox: string;
21534: var idHTTP: TIDHTTP;
21535: begin
21536:   result:= 'version not found';
21537:   if IsInternet then begin
21538:     idHTTP:= TIdHTTP.Create(NIL);
21539:     try
21540:       result:= idHTTP.Get(MXVERSIONFILE2);
21541:       result:= result[1]+result[2]+result[3]+result[4]+result[5];
21542:       if result = MBVER2 then begin
21543:         //output.Font.Style:= [fsbold];
21544:         //Speak(' A new Version '+vstr+' of max box is available! ');
21545:         result:= ('!!!! OK. You have latest Version: '+result+ ' available at '+MXSITE);
21546:       end;
21547:     //idhttp.get2('http://www.softwareschule.ch/maxbox.htm')
21548:     finally
21549:       idHTTP.Free
21550:     end;
21551:   end;
21552: end;
21553:
21554:
21555: /////////////////////////////////
21556: All maxBox Tutorials Table of Content 2014
21557: ///////////////////////////////
21558: Tutorial 00 Function-Coding (Blix the Programmer)
21559: Tutorial 01 Procedural-Coding
21560: Tutorial 02 OO-Programming
21561: Tutorial 03 Modular Coding
21562: Tutorial 04 UML Use Case Coding
21563: Tutorial 05 Internet Coding
21564: Tutorial 06 Network Coding
21565: Tutorial 07 Game Graphics Coding
21566: Tutorial 08 Operating System Coding
21567: Tutorial 09 Database Coding
21568: Tutorial 10 Statistic Coding
21569: Tutorial 11 Forms Coding
21570: Tutorial 12 SQL DB Coding
21571: Tutorial 13 Crypto Coding
21572: Tutorial 14 Parallel Coding
21573: Tutorial 15 Serial RS232 Coding
21574: Tutorial 16 Event Driven Coding
21575: Tutorial 17 Web Server Coding
21576: Tutorial 18 Arduino System Coding
21577: Tutorial 18_3 RGB LED System Coding
21578: Tutorial 19 WinCOM /Arduino Coding
21579: Tutorial 20 Regular Expressions RegEx
21580: Tutorial 21 Android Coding (coming 2013)
21581: Tutorial 22 Services Programming
21582: Tutorial 23 Real Time Systems
21583: Tutorial 24 Clean Code
21584: Tutorial 25 maxBox Configuration I+II
21585: Tutorial 26 Socket Programming with TCP
21586: Tutorial 27 XML & TreeView
21587: Tutorial 28 DLL Coding (available)
21588: Tutorial 29 UML Scripting (2014)
21589: Tutorial 30 Web of Things (2014)
21590: Tutorial 31 Closures (coming 2014)
21591: Tutorial 32 SQL Firebird (coming 2014)
21592: Tutorial 33 Oscilloscope (coming 2015)
21593: Tutorial 34 GPS Navigation (coming 2015)
21594: Tutorial 35 Web Box (available)
21595:
21596: Doc ref Docu for all Type Class and Const in maxbox_types.pdf
21597: using Docu for this file is maxbox_functions_all.pdf
21598: PEP - Pascal Education Program Lib Lab ShellHell
21599:
21600: http://stackoverflow.com/tags/pascalscript/hot
21601: http://www.jrsoftware.org/isihelp/index.php?topic=scriptfunctions
21602: http://sourceforge.net/projects/maxbox #locs:51620
21603: http://sourceforge.net/apps/mediawiki/maxbox
21604: http://www.blaisepascal.eu/
21605: https://github.com/maxkleiner/maxBox3.git
21606: http://www.heise.de/download/maxbox-1176464.html
21607: http://www.softpedia.com/get/Programming/Other-Programming-Files/maxbox.shtml
21608: https://www.facebook.com/pages/Programming-maxBox/166844836691703
21609: http://www.softwareschule.ch/arduino_training.pdf
21610: http://www.delphiarea.com
21611:
21612: All maxBox Examples List
21613: https://github.com/maxkleiner/maxBox3/releases
21614: ****
21615: 000_pas_baseconvert.txt 282_fadengraphik.txt
21616: 000_pas_baseconvert.txt_encrypt 283_SQL_API_messageTimeout.txt
21617: 000_pas_baseconvert.txt_decrypt 284_SysTools4.txt
21618: 001_1_pas_functest - Kopie.txt 285_MineForm_GR32.TXT

```

```

21619: 001_1_pas_functest.txt
21620: 001_1_pas_functest2.txt
21621: 001_1_pas_functest_clx2.txt
21622: 001_1_pas_functest_clx2_2.txt
21623: 001_1_pas_functest_openarray.txt
21624: 001_pas_lottogen.txt
21625: 001_pas_lottogen_template.txt
21626: 001_pas_lottogen.txtcopy
21627: 002_pas_russianroulette.txt
21628: 002_pas_russianroulette.txtcopy
21629: 002_pas_russianroulette.txtcopy_decrypt
21630: 002_pas_russianroulette.txtcopy_encrypt
21631: 003_pas_motion.txt
21632: 003_pas_motion.txtcopy
21633: 004_pas_search.txt
21634: 004_pas_search_replace.txt
21635: 004_search_replace_allfunctionlist.txt
21636: 005_pas_oodeign.txt
21637: 005_pas_shellink.txt
21638: 006_pas_oobatch.txt
21639: 007_pas_streamcopy.txt
21640: 008_EINMALEINS_FUNC.TXT
21641: 008_explanation.txt
21642: 008_pas_verwechselt.txt
21643: 008_pas_verwechselt_ibz_bern_func.txt
21644: 008_stack_ibz.TXT
21645: 009_pas_umrunner.txt
21646: 009_pas_umrunner_all.txt
21647: 009_pas_umrunner_componenttest.txt
21648: 009_pas_umrunner_solution.txt
21649: 009_pas_umrunner_solution_2step.txt
21650: 010_pas_oodeign_solution.txt
21651: 011_pas_puzzlepas_defect.txt
21652: 012_pas_umrunner_solution.txt
21653: 012_pas_umrunner_solution2.txt
21654: 013_pas_linenumber.txt
21655: 014_pas_primetest.txt
21656: 014_pas_primetest_first.txt
21657: 014_pas_primetest_sync.txt
21658: 015_pas_designbycontract.txt
21659: 015_pas_designbycontract_solution.txt
21660: 016_pas_searchrec.txt
21661: 017_chartgen.txt
21662: 018_data_simulator.txt
21663: 019_dez_to_bin.txt
21664: 019_dez_to_bin_grenzwert_ibz.txt
21665: 020_proc_feedback.txt
21666: 021_pas_symkey.txt
21667: 021_pas_symkey_solution.txt
21668: 022_pas_filestreams.txt
21669: 023_pas_find_searchrec.txt
21670: 023_pas_pathfind.txt
21671: 024_pas_TFileStream_records.txt
21672: 025_prime_direct.txt
21673: 026_pas_memorystream.txt
21674: 027_pas_shellexecute_beta.txt
21675: 027_pas_shellexecute_solution.txt
21676: 028_pas_dataset.txt
21677: 029_pas_assignfile.txt
21678: 029_pas_assignfile_dragndropexe.txt
21679: 030_palindrome_2.txt
21680: 030_palindrome_tester.txt
21681: 030_pas_recursion.txt
21682: 030_pas_recursion2.txt
21683: 031_pas_hashcode.txt
21684: 032_pas_crc_const.txt
21685: 033_pas_cipher.txt
21686: 033_pas_cipher_def.txt
21687: 033_pas_cipher_file_2_solution.txt
21688: 034_pas_soundbox.txt
21689: 035_pas_crcscript.txt
21690: 035_pas_CRCscript_modbus.txt
21691: 036_pas_includetest.txt
21692: 036_pas_includetest_basta.txt
21693: 037_pas_define_demo32.txt
21694: 038_pas_box_demonstrator.txt
21695: 039_pas_dllcall.txt
21696: 040_paspointer.txt
21697: 040_paspointer_old.txt
21698: 041_pasplotter.txt
21699: 041_pasplotter_plus.txt
21700: 042_pas_kgv_ggt.txt
21701: 043_pas_proceduretype.txt
21702: 044_pas_14queens_solwith14.txt
21703: 044_pas_8queens.txt
21704: 044_pas_8queens_sol2.txt
21705: 044_pas_8queens_solutions.txt
21706: 044_queens_performer.txt
21707: 044_queens_performer2.txt

285_MineForm_GR32main.TXT
285_MineForm_GR32mainsolution.TXT
285_MineForm_propas.TXT
285_MineForm_propas2.TXT
285_minesweeper2.TXT
285_Patterns_process.txt
286_colormixer_jpeg_charcounter.txt
286_colormixer_jpeg_charcounter2.txt
287_eventhandling.txt
287_eventhandling2.txt
287_eventhandling2_negpower.txt
288_bitblt.txt
288_bitblt_resize.txt
289_regression.txt
289_regression2.txt
290_bestofbox.txt
290_bestofbox2.txt
290_bestofbox3.txt
291_3sort_visual_thread.txt
292_refactoring2.txt
293_bold_utils.txt
293_ib_utils.txt
293_ib_utils_timetest.txt
294_maxcalc_demo.txt
294_maxcalc_demo2.txt
295_easter_calendar.txt
295_easter_calendar2.txt
295_easter_combobox.txt
297_atomimage.txt
297_atomimage2.txt
297_atomimage3.txt
297_atomimage4.txt
297_maxonmotor.TXT
297_maxon_atomimage9.txt
298_bitblt_animation.txt
298_bitblt_animation2.txt
298_bitblt_animation3.txt
298_bitblt_animation4.txt
298_bitblt_animation4_screensaver.txt
298_bitblt_animation5_screensaver.txt
299_animation.txt
299_animationmotor_arduino.txt
299_animation_formprototype.txt
299_realtimeclock_arduino.txt
299_realtimeclock_arduino2.txt
300_treeview.txt
300_treeview_test.txt
300_treeview_test2.txt
300_treeview_test3.txt
301_LED_Arduino3.txt
301_led_arduino3_simple.txt
301_led_arduino3_simplecode.txt
301_log_arduino.txt
301_log_arduino2.txt
301_SQL_DBfirebird3.txt
301_SQL_DBfirebird4.txt
302_LCLActivity_java.txt
302_LED_DataLogger.txt
303_Android_LCLActivity_java.txt
303_webserver.txt
303_webserver2.txt
303_webserver_alldocs2.txt
303_webserver_alldocs2_tester.txt
303_webserver_minimal.txt
303_webserver_simple.txt
304_st_system.txt
305_indy_elizahttpserver.TXT
305_indy_elizahttpserver2.TXT
305_indy_elizahttpserver3.TXT
305_indy_elizahttpserver4file.TXT
305_webserver_arduino.txt
305_webserver_arduino2.txt
305_webserver_arduino3.txt
305_webserver_arduino3ibz.txt
305_webserver_arduino3ibz_rgb_led.txt
305_webserver_arduino3test.txt
306_SPS_http_command.txt
307_all_booleanlogic.txt
308_bitbox3.txt
308_bitbox3_exec.txt
308_boolean_animation.txt
308_boolean_animation2.txt
309_regex_power.txt
309_regex_powertester2.txt
309_regex_powertester3.txt
310_regex_decorator.TXT
312_ListView.txt
313_dmath_dll.txt
314_fundamentals4_tester.TXT

```

```

21708: 044_pas_queens_performer2tester.txt
21709: 045_pas_listhandling.txt
21710: 046_pas_records.txt
21711: 047_pas_modulal0.txt
21712: 048_pas_romans.txt
21713: 049_pas_ifdemo.txt
21714: 049_pas_ifdemo_BROKER.txt
21715: 050_pas_primetest2.txt
21716: 050_pas_primetester_thieves.txt
21717: 050_program_starter.txt
21718: 050_program_starter_performance.txt
21719: 051_pas_findtext_solution.txt
21720: 052_pas_text_as_stream.txt
21721: 052_pas_text_as_stream_include.txt
21722: 053_pas_singleton.txt
21723: 054_pas_speakpassword.txt
21724: 054_pas_speakpassword2.txt
21725: 054_pas_speakpassword_searchtest.txt
21726: 055_pas_factorylist.txt
21727: 056_pas_demeter.txt
21728: 057_pas_dirfinder.txt
21729: 058_pas_filefinder.txt
21730: 058_pas_filefinder_pdf.txt
21731: 058_pas_filefinder_screview.txt
21732: 058_pas_filefinder_screview2.txt
21733: 058_pas_filefinder_screview3.txt
21734: 059_pas_timertest.txt
21735: 059_pas_timertest_2.txt
21736: 059_pas_timertest_time_solution.txt
21737: 059_timerobject_starter2.txt
21738: 059_timerobject_starter2_ibz2_async.txt
21739: 059_timerobject_starter2_uml.txt
21740: 059_timerobject_starter2_uml_main.txt
21741: 059_timerobject_starter4_ibz.txt
21742: 060_pas_datefind.txt
21743: 060_pas_datefind_exceptions2.txt
21744: 060_pas_datefind_exceptions_CHECKTEST.txt
21745: 060_pas_datefind_fulltext.txt
21746: 060_pas_datefind_plus.txt
21747: 060_pas_datefind_plus_mydate.txt
21748: 061_pas_randomwalk.txt
21749: 061_pas_randomwalk_plus.txt
21750: 062_paskorrelation.txt
21751: 063_pas_calculateform.txt
21752: 063_pas_calculateform_2list.txt
21753: 064_pas_timetest.txt
21754: 065_pas_bitcounter.txt
21755: 066_pas_eliza.txt
21756: 066_pas_eliza_include_sol.txt
21757: 067_pas_morse.txt
21758: 068_pas_piezo_sound.txt
21759: 069_LED_Matrix_R1_3_6_NV_PSchaer.TXT
21760: 069_my_LEDBOX.TXT
21761: 069_pas_ledmatrix.txt
21762: 069_pas_LEDMATRIX_Alphabet.txt
21763: 069_pas_LEDMATRIX_Alphabet_run.txt
21764: 069_pas_LEDMATRIX_Alphabet_tester.txt
21765: 069_PAS_LEDMATRIX_COLOR.TXT
21766: 069_pas_ledmatrix_fixedit.txt
21767: 069_pas_LEDMATRIX_soundbox.txt
21768: 069_pas_LEDMATRIX_soundbox2.txt
21769: 069_Richter_MATRIX.TXT
21770: 070_pas_functionplot.txt
21771: 070_pas_functionplotter2.txt
21772: 070_pas_functionplotter2_mx4.txt
21773: 070_pas_functionplotter2_tester.txt
21774: 070_pas_functionplotter3.txt
21775: 070_pas_functionplotter4.txt
21776: 070_pas_functionplotter_digital.txt
21777: 070_pas_functionplotter_elliptic.txt
21778: 070_pas_function_helmholtz.txt
21779: 070_pas_textcheck_experimental.txt
21780: 071_pas_graphics.txt
21781: 071_pas_graphics_drawsym.txt
21782: 071_pas_graphics_drawsym_save.txt
21783: 071_pas_graphics_random.txt
21784: 072_pas_fractals.txt
21785: 072_pas_fractals_2.txt
21786: 072_pas_fractals_blackhole.txt
21787: 072_pas_fractals_performace.txt
21788: 072_pas_fractals_performance_new.txt
21789: 072_pas_fractals_performace_sharp.txt
21790: 072_pas_fractals_performance.txt
21791: 072_pas_fractals_performance_mx4.txt
21792: 073_pas_forms.txt
21793: 074_pas_chartgenerator.txt
21794: 074_pas_chartgenerator_solution.txt
21795: 074_pas_chartgenerator_solution_back.txt
21796: 074_pas_charts.txt

315_funcplot_dmath.TXT
316_cfileutils_cdatetime_tester.TXT
317_excel_export_tester.TXT
318_excel_export.TXT
318_excel_export2.TXT
318_excel_export3.TXT
318_excel_export3_tester.TXT
319_superfunctions_math.TXT
319_superfunctions_mathdefect.TXT
320_superfunctions.TXT
320_superfunctions2.TXT
321_SQL_Excel.txt
321_SQL_Excel2.txt
321_SQL_Excel_Export.txt
321_SQL_ExportExec.txt
321_SQL_ExportTest.txt
321_SQL_SAS_tester3.txt
321_SQL_SAS_tester3_selfcompile.txt
321_SQL_SAS_tester3_selfcompile2.txt
321_SQL_SAS_tester4.txt
321_SQL_SAS_updater.txt
322_timezones.TXT
323_datefind_fulltext_search.txt
323_datefind_fulltext_searchtester.txt
324_interfacenavi.TXT
325_ampelsteuerung.txt
325_analogclock.txt
326_world_analogclock.txt
326_world_analogclock2.txt
327_atomimage_clock.txt
328_starfield.txt
329_starfield2.txt
330_myclock.txt
330_myclock2.txt
331_SQL_DBfirebird4.txt
332_jprofiler.txt
332_jprofiler_form.txt
332_jprofiler_form2.txt
333_querybyexample.txt
333_querybyexample2.txt
334_jvutils_u.txt
335_atomimage5.txt
335_atomimage6.txt
335_atomimage7.txt
336_digiclock.txt
336_digiclock2.txt
336_digiclock2test.txt
336_digiclock3.txt
337_4games.txt
337_4games_inone.txt
338_compress.txt
338_compress2.txt
339_ntfs.txt
340_docudtype.txt
340_logsimulation.txt
340_logsimulation2.txt
340_soundControltype.txt
341_blix_clock.txt
341_blix_clock2.txt
341_blix_clock_tester.txt
342_set_enumerator.txt
343_dice2.txt
344_pe_header.txt
344_pe_header2.txt
345_velocity.txt
346_conversions.txt
347_pictureview.txt
348_dualistview.txt
349_biginteger.txt
350_parserform.txt
351_chartform.txt
351_chartform2.txt
351_chartform3.txt
352_array_unittest.txt
353_smtp_email.txt
353_smtp_email2.txt
354_josephus.txt
355_life_of_PI.txt
356_3D_printer.txt
357_fplot.TXT
358_makesound.txt
359_charsetrules.TXT
360_allobjects.TXT
360_JvPaintFX.TXT
361_heartbeat_wave.TXT
362_maxonmotor2.TXT
363_compress_services.txt
363_compress_services2.txt
364_pdf_services.txt

```

```

21797: 075_bitmap_Artwork2.txt
21798: 075_pas_bitmappuzzle.txt
21799: 075_pas_bitmappuzzle24.prod.txt
21800: 075_pas_bitmappuzzle2_prod.txt
21801: 075_pas_bitmappuzzle3.txt
21802: 075_pas_bitmapsolve.txt
21803: 075_pas_bitmap_Artwork.txt
21804: 075_pas_puzzlepas_solution.txt
21805: 076_pas_3dcube.txt
21806: 076_pas_circle.txt
21807: 077_pas_mmshow.txt
21808: 078_pas_pi.txt
21809: 079_pas_3dcube_animation.txt
21810: 079_pas_3dcube_animation4.txt
21811: 079_pas_3dcube_plus.txt
21812: 080_pas_hanoi.txt
21813: 080_pas_hanoi2.txt
21814: 080_pas_hanoi2_file.txt
21815: 080_pas_hanoi2_sol.txt
21816: 080_pas_hanoi2_tester.txt
21817: 080_pas_hanoi2_tester_fast.txt
21818: 080_pas_hanoi3.txt
21819: 081_pas_chartist2.txt
21820: 082_pas_biorhythmus.txt
21821: 082_pas_biorhythmus_solution.txt
21822: 082_pas_biorhythmus_solution_3.txt
21823: 082_pas_biorhythmus_test.txt
21824: 083_pas_GITARRE.txt
21825: 083_pas_soundbox_tones.txt
21826: 084_pas_waves.txt
21827: 085_mxsinus_logo.txt
21828: 085_sinus_plot_waves.txt
21829: 086_pas_graph_arrow_heart.txt
21830: 087_bitmap_loader.txt
21831: 087_pas_bitmap_solution.txt
21832: 087_pas_bitmap_solution2.txt
21833: 087_pas_bitmap_subimage.txt
21834: 087_pas_bitmap_test.txt
21835: 088_pas_soundbox2_mp3.txt
21836: 088_pas_soundbox_mp3.txt
21837: 088_pas_sphere_2.txt
21838: 089_pas_gradient.txt
21839: 089_pas_maxland2.txt
21840: 090_pas_sudoku4.txt
21841: 090_pas_sudoku4_2.txt
21842: 091_pas_cube4.txt
21843: 092_pas_statistics4.txt
21844: 093_variance.txt
21845: 093_variance_debug.txt
21846: 094_pas_daysold.txt
21847: 094_pas_stat_date.txt
21848: 095_pas_ki_simulation.txt
21849: 096_pas_geisen_problem.txt
21850: 096_pas_montyhall_problem.txt
21851: 097_lotto_proofofconcept.txt
21852: 097_pas_lottocombinations_beat_plus.txt
21853: 097_pas_lottocombinations_beat_plus2.txt
21854: 097_pas_lottocombinations_universal.txt
21855: 097_pas_lottosimulation.txt
21856: 098_pas_chartgenerator_plus.txt
21857: 099_pas_3D_show.txt
21858: 200_big_numbers.txt
21859: 200_big_numbers2.txt
21860: 201_streamload_xml.txt
21861: 202_systemcheck.txt
21862: 203_webservice_simple_intftester.txt
21863: 204_webservice_simple.txt
21864: 205_future_value_service.txt
21865: 206_DTD_string_functions.txt
21866: 207_ibz2_async_process.txt
21867: 208_crc32_hash.txt
21868: 209_cryptohash.txt
21869: 210_public_private.txt
21870: 210_public_private_cryptosystem.txt
21871: 211_wipe_pattern.txt
21872: 211_wipe_pattern2.txt
21873: 211_wipe_pattern_solution.txt
21874: 212_pas_statisticmodule4.TXT
21875: 212_pas_statisticmoduletxt.TXT
21876: 212_statisticmodule4.txt
21877: 213_pas_BBP_Algo.txt
21878: 214_mxdocudemo.txt
21879: 214_mxdocudemo2.txt
21880: 214_mxdocudemo3.txt
21881: 215_hints_test.TXT
21882: 216_warnings_test.TXT
21883: 217_pas_heartbeat.txt
21884: 218_biorhythmus_studio.txt
21885: 219_cipherbox.txt

365_memorystream.txt
365_memorystream2.txt
365_memorystream_test.txt
365_U_HexView.txt
366_mp3player.txt
366_mp3player2.txt
366_mp3player2_themestest.txt
367_silvi_player_widgets.txt
367_silvi_player_widgets2.txt
367_widgets.txt
368_configuration_demo.txt
369_macro_demo.txt
370_callback2grid.TXT
370_richedit.txt
370_richedit_highlight.txt
370_synedit.txt
370_synedit2.txt
370_synedit2_mxtester.txt
370_synedit2_mxtester2.txt
371_maxbook_v4tester.txt
372_stackibz2_memoryalloc.TXT
372_synedit_export.txt
373_batman.txt
373_fractals_tvout.txt
374_realtime_random.txt
374_realtime_random2.txt
374_realtime_randomtest.txt
374_realtime_randomtest2.txt
375_G9_musicbox.txt
376_collections_list.txt
377_simpleXML.txt
377_smartXML.txt
377_smartXMLWorkshop.txt
377_smartXMLWorkshop2.txt
378_queryperformance3.txt
378_REST1.txt
378_REST2.txt
379_timefunc.txt
379_timefuncTesterfilemon.txt
380_coolfunc.txt
380_coolfunc2.txt
380_coolfunc_tester.txt
381_bitcoin_simulation.txt
382_GRMath.TXT
382_GRMath_PI_Proof.TXT
382_GRMath_Riemann.TXT
383_MDAC_DCOM.txt
384_TeamViewerID.TXT
386_InternetRadio.TXT
387_fulltextfinder.txt
387_fulltextfinder_cleancode.txt
387_fulltextfinder_fast.txt
387_fulltext_getscripttest.txt
388_TCPServerSock.TXT
388_TCPServerSock2.TXT
388_TCPServerSockClient.TXT
389_TAR_Archive.TXT
389_TAR_Archive_test.TXT
389_TAR_Archive_test2.TXT
390_Callback3.TXT
390_Callback3Rec.TXT
390_CallbackClean.TXT
390_StringlistHTML.TXT
391_ToDo_List.TXT
392_BarcodE.TXT
392_BarcodE2.TXT
392_BarcodE23.TXT
392_BarcodE2scholz.TXT
392_BarcodE3scholz.TXT
393_QRCode.TXT
393_QRCode2.TXT
393_QRCode2Direct.TXT
393_QRCode2DirectIndy.TXT
393_QRCode2Direct_detlef.TXT
393_QRCode3.TXT
394_networkgraph.TXT
394_networkgraph_depwalkutilstest.TXT
394_networkgraph_depwalkutilstest2.TXT
395_USBController.TXT
396_Sort.TXT
397_Hotlog.TXT
397_Hotlog2.TXT
398_ustrings.txt
399_form_templates.txt
400_fploottchart.TXT
400_fploottchart2.TXT
400_fploottchart2teetest.TXT
400_QRCodeMarket.TXT
401_tfilerun.txt

```

```

21886: 219_crypt_source_comtest_solution.TXT
21887: 220_cipherbox_form.txt
21888: 220_cipherbox_form2.txt
21889: 221_bcd_explain.txt
21890: 222_memoform.txt
21891: 223_directorybox.txt
21892: 224_dialogs.txt
21893: 225_sprite_animation.txt
21894: 226_ASCII_Grid2.TXT
21895: 227_animation.txt
21896: 227_animation2.txt
21897: 228_android_calendar.txt
21898: 229_android_game.txt
21899: 229_android_game_tester.txt
21900: 230_DataProvider.txt
21901: 230_DataSetProvider.txt
21902: 230_DataSetXMLBackupScholz.txt
21903: 231_DBGrid_access.txt
21904: 231_DBGrid_XMLaccess.txt
21905: 231_DBGrid_XMLaccess2.txt
21906: 231_DBGrid_XMLaccess_locatetester.txt
21907: 231_DBGrid_XML_CDS_local.txt
21908: 232_outline.txt
21909: 232_outline_2.txt
21910: 233_modular_form.txt
21911: 234_debugoutform.txt
21912: 235_fastform.TXT
21913: 236_componentpower.txt
21914: 236_componentpower_back.txt
21915: 237_pas_4forms.txt
21916: 238_lottogen_form.txt
21917: 239_pas_sierpinski.txt
21918: 239_pas_sierpinski2.txt
21919: 240_unitGlobal_tester.txt
21920: 241_db3_sql_tutorial.txt
21921: 241_db3_sql_tutorial12.txt
21922: 241_db3_sql_tutorial2fix.txt
21923: 241_db3_sql_tutorial3.txt
21924: 241_db3_sql_tutorial3connect.txt
21925: 241_db3_sql_tutorial3_ftptest.txt
21926: 241_RTL_SET2.txt
21927: 241_RTL_SET2_tester.txt
21928: 242_Component_Control.txt
21929: 243Tutorial_loader.txt
21930: 244_script_loader_loop.txt
21931: 245_formapp2.txt
21932: 245_formapp2_tester.txt
21933: 245_formapp2_testerX.txt
21934: 246_httpapp.txt
21935: 247_datecalendar.txt
21936: 248_ASCII_Grid2_sorted.TXT
21937: 249_picture_grid.TXT
21938: 250_tipsandtricks2.txt
21939: 250_tipsandtricks3.txt
21940: 250_tipsandtricks3api.txt
21941: 250_tipsandtricks3_admin_elevation.txt
21942: 250_tipsandtricks3_tester.txt
21943: 250_tipsandtricks4_tester.txt
21944: 250_tipsandtricks4_tester2.txt
21945: 251_compare_noise_gauss.txt
21946: 251_whitenoise.txt
21947: 251_whitenoise2.txt
21948: 252_hilbert_turtle.txt
21949: 252_pas_hilbert.txt
21950: 253_opearatingsystem3.txt
21951: 254_dynarrays.txt
21952: 255_einstein.txt
21953: 256_findconsts_of_EXE.txt
21954: 256_findfunctions2_of_EXE.txt
21955: 256_findfunctions2_of_EXBAverp.txt
21956: 256_findfunctions2_of_EXEspec.txt
21957: 256_findfunctions3.txt
21958: 256_findfunctions_of_EXE.txt
21959: 257_AES_Cipher.txt
21960: 258_AES_cryptobox.txt
21961: 258_AES_cryptobox2.txt
21962: 258_AES_cryptobox2_passdlg.txt
21963: 259_AES_crypt_directory.txt
21964: 260_sendmessage_2.TXT
21965: 260_sendmessage_beta.TXT
21966: 261_probability.txt
21967: 262_mxoutputdemo4.txt
21968: 263_async_sound.txt
21969: 264_vclutils.txt
21970: 264_VCL_utils2.txt
21971: 265_timer_API.txt
21972: 266_serial_interface.txt
21973: 266_serial_interface2.txt
21974: 266_serial_interface3.txt
402_richedit2.txt
403_outlookspy.txt
404_simplebrowser.txt
405_datefinder_today.txt
406_portscan.txt
407_indydemo.txt
408_testroboter.txt
409_excel_control.txt
410_keyboardevent.txt
411_json_test.txt
412_Zeosutils.txt
413_listview2.txt
414_avrdude_flash.txt
415_avrdude_writehex.txt
416_sonar_startscriptEKON.TXT
416_sonar_startscriptEKON_reporting.TXT
417_GRMath_PI_Proof2.TXT
418_functional_paradigm.txt
419_archimedes_spiral.txt
419_archimedes_spiral2.txt
420_archimedes_arduino.txt
420_Lissajous.txt
421_PI_Power.TXT
421_PI_Power2.TXT
422_world_bitboxx.txt
423_game_of_life.TXT
423_game_of_life2.TXT
423_game_of_life3.TXT
423_game_of_life3_test.TXT
423_game_of_life4.TXT
423_game_of_life4_kryptum.TXT
424_opengl_tester.txt
425_reversi_game.txt
426_IBUtils.TXT
427_IBDatabase.TXT
428_SortGrid.TXT
429_fileclass.txt
430_fileoperation.txt
430_fileoperation_tester.txt
431_performance_index.txt
432_shortstring_routines.txt
433_video_avicap.txt
433_video_avicap2.txt
434_GSM_module.TXT
435_httpcommon.txt
436_GraphicSplitter.txt
436_GraphicSplitter_form.txt
436_GraphicSplitter_form2.txt
436_teetest_screen.TXT
436_teetest_screen2.TXT
437_WinAPItop.txt
437_WinAPItop_Firebirdtester.txt
438_OvcInternational.txt
439_AsyncFreeDemo.txt
439_AsyncFreeDemoForm.txt
440_DLL_Tutor.txt
440_DLL_Tutor2.txt
440_XML_Tutor.txt
440_XML_Tutor2.txt
441_make_app.txt
442_arduino_rgb_led.txt
443_webserver_arduino_rgb_light.txt
443_webserver_arduino_rgb_light4.txt
444_webserver_arduino3ibz_rgb_led_basta.txt
445_datagrid.txt
445_datagrid2.txt
445_datagrid_android_arduino.txt
446_arduino_timer.txt
447_patternFrm_mx3.txt
448_Synapse.txt
448_Synapse2.txt
449_dweb_start_tester.txt
450_Synapse_HTTPS.txt
450_Synapse_Mime.txt
450_Synapse_ScanPing.txt
451_ocx_player.txt
451_OCX_WinPlayer2.txt
452_dbtreeview.txt
452_dbtreeview2.txt
453_stdfuncs.txt
454_fileStream.txt
455_functionfun.txt
455_functionfun2.txt
455_functionfun2_test.txt
457_ressource_grid.txt
458_atomimageX.txt
459_cindyfunc.txt
459_cindyfunc2.txt
460_TopTenFunctions.txt

```

```
21975: 267_ackermann_rec.txt          461_sqlform_calwin.txt
21976: 267_ackermann_variants.txt    462_caesarcipher.txt
21977: 268_DBGrid_tree.txt          463_global_exception.txt
21978: 269_record_grid.TXT          464_function_procedure.txt
21979: 270_Jedi_FunctionPower.txt   464_function_procedure2.txt
21980: 270_Jedi_FunctionPowertester.txt 464_function_procedure3.txt
21981: 271_closures_study.txt      465_U_HexView.txt
21982: 271_closures_study_workingset2.txt 466_moon.txt
21983: 272_pas_function_show.txt    466_moon_inputquery.txt
21984: 273_pas_function_show2.txt   4671_cardmagic.txt
21985: 274_library_functions.txt   467_helmholtz_graphic.txt
21986: 275_turtle_language.txt     468_URLMon.txt
21987: 275_turtle_language_save.txt 468_URLMon2.txt
21988: 276_save_algo.txt          469_formarrow.txt
21989: 276_save_algo2.txt         469_formarrow_datepicker.txt
21990: 277_functionsfor39.txt     469_formarrow_datepicker_ibz_result.txt
21991: 278_DB_Dialogs.TXT        469_ibzresult.txt
21992: 279_hexer2.TXT            470_DFFUtils_compiled.txt
21993: 279_hexer2macro.TXT       470_DFFUtils_ScrollingLED.txt
21994: 279_hexer2macroback.TXT    470_Oscilloscope.txt
21995: 280_UML_process.txt       470_Oscilloscope_code.txt
21996: 280_UML_process_knabe2.txt 471_cardmagic.txt
21997: 280_UML_process_knabe3.txt 471_cardmagic2.txt
21998: 280_UML_process_TIM_Botzenhardt.txt 472_allcards.TXT
21999: 280_UML_TIM_Seitz.txt     473_comboset.txt
22000: 281_picturepuzzle.txt     474_wakeonlan.txt
22001: 281_picturepuzzle2.txt    474_wakeonlan2.txt
22002: 281_picturepuzzle3.txt    476_getscripttest.txt
22003: 281_picturepuzzle4.txt    477_filenameonly.txt
22004: 479_inputquery.txt        480_regex_pathfinder.txt
22005: 480_regex_pathfinder2.txt  481_processList.txt
22006: 482 processPipe.txt       482 processPipeGCC.txt
22007:
22008:
22009: Web Script Examples:
22010:
22011: http://www.softwareschule.ch/examples/performer.txt';
22012: http://www.softwareschule.ch/examples/turtle.txt';
22013: http://www.softwareschule.ch/examples/SQLExport.txt';
22014: http://www.softwareschule.ch/examples/Richter.txt';
22015: http://www.softwareschule.ch/examples/checker.txt';
22016: http://www.softwareschule.ch/examples/demoscript.txt';
22017: http://www.softwareschule.ch/examples/ibzresult.txt';
22018: http://www.softwareschule.ch/examples/performindex.txt
22019: http://www.softwareschule.ch/examples/processlist.txt
22020: ----- bigbitbox code_cleared_checked-----
```