```
 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: 18788864 V3.9.9.88 March 2014 To Wirth EKON/BASTA 18 2014
10: ***************Now the Funclist*****************************************
11: Funclist Function : 11300 //10766 //10165 (7648)
12: ***************Now the Proclist*****************************************
13: Proclist Procedure Size is: 7289  //6792 //6401 //4752 (4552)
14: ***************Now Constructors*****************************************
15: Constructlist Constructor Size is: 1181 //995 //777 (689 /513 /494)
16: def head:max: maXbox7: 07.02.2014 19:11:55
17: file E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt
18: file: maxbox_extract_funclist399_.txt  (sort function list)
19: --------------------------------------------------------------------------------
20: Funclist total Size all is: 19770! Constructor, Function and Procedure
21: AExtraxt of EXE Functions of maxbox3.exe, locs of file = 18700
22: ASize of EXE:  18788864 (16586240) (13511680) (13023744)
23: SHA1 Hash of maXbox 3.9.9.88: 119533C0725A9B9B2919849759AA2F6298EBFF28
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:Pointer):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: TTextLineBreakStyle', '(tlbsLF, tlbsCRLF)
109: Function AdjustLineBreaks(const S: string; Style: TTextLineBreakStyle): string;
110: Function AllData : string
111: function AllocMemCount: integer;
112: function AllocMemSize: integer;
113: Function AllocPatternBitmap( BkColor, FgColor : TColor) : TBitmap
114: Function AllowRegKeyForEveryone( Key : HKEY; Path : string) : Boolean
115: Function AlphaComponent( const Color32 : TColor32) : Integer
116: Function AlphaSort : Boolean
117: Function AlphaSort( ARecurse : Boolean) : Boolean
118: Function AnsiCat( const x, y : AnsiString) : AnsiString
119: Function AnsiCompareFileName( S1, S2 : string) : Integer
120: function AnsiCompareFileName(const S1: string; const S2: string): Integer)
121: Function AnsiCompareStr( S1, S2 : string) : Integer
122: function AnsiCompareStr(const S1: string; const S2: string): Integer;)
123: Function AnsiCompareText( S1, S2 : string) : Integer
124: function AnsiCompareText(const S1: string; const S2: string): Integer;)
125: Function AnsiContainsStr( const AText, ASubText : string) : Boolean
126: Function AnsiContainsText( const AText, ASubText : string) : Boolean
127: Function AnsiCopy( const src : AnsiString; index, count : Integer) : AnsiString
128: Function AnsiDequotedStr( S : string; AQuote : Char) : string
129: Function AnsiEndsStr( const ASubText, AText : string) : Boolean
130: Function AnsiEndsText( const ASubText, AText : string) : Boolean
131: Function AnsiExtractQuotedStr( var Src : PChar; Quote : Char) : string
132: function AnsiExtractQuotedStr(var Src: PChar; Quote: Char): string)
133: Function AnsiIndexStr( const AText : string; const AValues : array of string) : Integer
134: Function AnsiIndexText( const AText : string; const AValues : array of string) : Integer
135: Function AnsiLastChar( S : string) : PChar
136: function AnsiLastChar(const S: string): PChar)
137: Function AnsiLeftStr( const AText : AnsiString; const ACount : Integer) : AnsiString
138: Function AnsiLowerCase( S : string) : string
139: Function AnsiLowercase(s : String) : String;
140: Function AnsiLowerCaseFileName( S : string) : string
141: Function AnsiMatchStr( const AText : string; const AValues : array of string) : Boolean
142: Function AnsiMatchText( const AText : string; const AValues : array of string) : Boolean
143: Function AnsiMidStr( const AText : AnsiString; const AStart, ACount : Integer) : AnsiString
144: Function AnsiPos( const src, sub : AnsiString) : Integer
145: Function AnsiPos( Substr, S : string) : Integer
146: function AnsiPos(const Substr: string; const S: string): Integer;)
147: Function AnsiQuotedStr( S : string; Quote : Char) : string
148: Function AnsiReplaceStr( const AText, AFromText, AToText : string) : string
149: Function AnsiReplaceText( const AText, AFromText, AToText : string) : string
150: Function AnsiResemblesText( const AText, AOther : string) : Boolean
151: Function AnsiReverseString( const AText : AnsiString) : AnsiString
152: Function AnsiRightStr( const AText : AnsiString; const ACount : Integer) : AnsiString
153: function AnsiSameCaption(const Text1: string; const Text2: string): Boolean)
154: Function AnsiSameStr( S1, S2 : string) : Boolean
155: function AnsiSameStr(const S1: string; const S2: string): Boolean)
156: Function AnsiSameText( const S1, S2 : string) : Boolean
157: Function AnsiSameText( S1, S2 : string) : Boolean
158: function AnsiSameText(const S1: string; const S2: string): Boolean)
159: Function AnsiStartsStr( const ASubText, AText : string) : Boolean
160: Function AnsiStartsText( const ASubText, AText : string) : Boolean
161: Function AnsiStrComp( S1, S2 : PChar) : Integer
162: function AnsiStrComp(S1: PChar; S2: PChar): Integer)
163: Function AnsiStrIComp( S1, S2 : PChar) : Integer
164: function AnsiStrIComp(S1: PChar; S2: PChar): Integer)
165: Function AnsiStrLastChar( P : PChar) : PChar
166: function AnsiStrLastChar(P: PChar): PChar)
167: Function AnsiStrLComp( S1, S2 : PChar; MaxLen : Cardinal) : Integer
168: Function 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(I: Longint): Boolean;
208: Function AspectRatio(aWidth, aHeight: Integer): String;
209: Function AsWideString( Field : TField) : WideString
210: Function AtLeast( ACount : Integer) : Boolean
211: Function AttemptToUseSharedMemoryManager : Boolean
212: Function Authenticate : Boolean
213: Function AuthenticateUser( const AUsername, APassword : String) : Boolean
214: Function Authentication : String
215: Function BatchMove( ASource : TBDEDataSet; AMode : TBatchMode) : Longint
216: Function BcdCompare( const bcd1, bcd2 : TBcd) : Integer
217: Function BcdFromBytes( const AValue : TBytes) : TBcd
218: Function BcdPrecision( const Bcd : TBcd) : Word
219: Function BcdScale( const Bcd : TBcd) : Word
220: Function BcdToBytes( const Value : TBcd) : TBytes
221: Function BCDToCurr( const BCD : TBcd; var Curr : Currency) : Boolean
222: Function BcdToDouble( const Bcd : TBcd) : Double
223: Function BcdToInteger( const Bcd : TBcd; Truncate : Boolean) : Integer
224: Function BcdToStr( const Bcd : TBcd) : string;
225: Function BcdToStrF(const Bcd : TBcd; Format: TFloatFormat; const Precision, Digits:Integer):string
226: function beep2(dwFreq, dwDuration: integer): boolean;
227: Function BeginPeriod( const Period : Cardinal) : Boolean
228: Function BeginTrans : Integer
229: Function BeginTransaction : TDBXTransaction;
230: Function BeginTransaction1( Isolation : TDBXIsolation) : TDBXTransaction;
231: function BigMulu(aone, atwo: string): string;
232: function BigNumber(aone, atwo: string): string;
233: function BigExp(aone, atwo: string): string;
234: function BigMul(aone, atwo: string): string;
235: function BigAdd(aone, atwo: string): string;
236: function BigSub(aone, atwo: string): string;
237: function BigFactorial(aone: string): string;
238: Function BinaryToDouble( ABinary : string; DefValue : Double) : Double
239: Function BinomialCoeff( N, R : Cardinal) : Float
240: function BinominalCoefficient(n, k: Integer): string;
241: Function BinStrToInt( const ABinary : String) : Integer
242: Function BinToByte(Binary: String): Byte;
243: function BinToHex2(Binary: String): string;
244: function BinToInt(Binary: String): Integer;
245: Function BinToChar(St: String): Char;
246: Function BinToStr(ans: string): string;
247: Function BitBlt(hdcDest:HDC;nXDest,nYDest,nWidth,nHeigh:Int;hdcSrc:HDC;nXSrc,nYSrc:Int;dwRop:DWORD):Bool;
248: Function BitmapsAreIdentical( ABitmap1, ABitmap2 : TBitmap) : Boolean
249: Function BitsHighest( X : Byte) : Integer;
250: Function BitsHighest1( X : ShortInt) : Integer;
251: Function BitsHighest2( X : SmallInt) : Integer;
252: Function BitsHighest3( X : Word) : Integer;
253: Function BitsHighest4( X : Integer) : Integer;
254: Function BitsHighest5( X : Cardinal) : Integer;
255: Function BitsHighest6( X : Int64) : Integer;
256: Function BitsLowest( X : Byte) : Integer;
257: Function BitsLowest1( X : Shortint) : Integer;
258: Function BitsLowest2( X : Smallint) : Integer;
259: Function BitsLowest3( X : Word) : Integer;
260: Function BitsLowest4( X : Cardinal) : Integer;
261: Function BitsLowest5( X : Integer) : Integer;
262: Function BitsLowest6( X : Int64) : Integer;
263: Function BitsNeeded( const X : Byte) : Integer;
264: Function BitsNeeded1( const X : Word) : Integer;
265: Function BitsNeeded2( const X : Integer) : Integer;
266: Function BitsNeeded3( const X : Int64) : Integer;
267: Function BlueComponent( const Color32 : TColor32) : Integer
```

```
268: Function BooleanToInteger( const Pb : Boolean) : Integer
269: Function BoolToStr(B: Boolean; UseBoolStrs: Boolean): string)
270: Function BoolToStr1(value : boolean) : string;
271: Function booltoint( aBool : Boolean) : LongInt
272: Function inttobool( aInt : LongInt) : Boolean
273: Function Bounds( ALeft, ATop, AWidth, AHeight : Integer) : TRect
274: function Bounds(ALeft, ATop, AWidth, AHeight: Integer): TRect)
275: Function BreakApart( BaseString, BreakString : string; StringList : 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 CharIsInEOF( const AString : string; ACharPos : Integer) : Boolean
327: Function CharIsInSet( const AString : string; const ACharPos : Integer; const ASet : String) : Boolean
328: Function CharLength( S : String; Index : Integer) : Integer
329: Function CharRange( const AMin, AMax : Char) : String
330: function CharsetToIdent(Charset: Longint; var Ident: string): Boolean)
331: Function CharToBin(vChr: Char): String;
332: Function CharNext(lpsz: PChar): PChar; stdcall;
333: Function CharToByteIndex( S : string; Index : Integer) : Integer
334: Function CharToByteIndex(const S: string; Index: Integer): Integer)
335: Function CharToByteLen( S : string; MaxLen : Integer) : Integer
336: function CharToByteLen(const S: string; MaxLen: Integer): Integer)
337: Function CharToHex(const APrefix : String; const cc : Char) : shortstring;
338: function CharToHexStr(Value: char): string);
339: function CharToOem(ins, outs: PChar):boolean;
340: function CharToUniCode(Value: Char): string;
341: Function CheckMenuDropdown : Boolean
342: Function CheckMessages : longint
343: Function CheckOpen( Status : DBIResult) : Boolean
344: Function CheckPassword( const APassword : String) : Boolean
345: Function CheckResponse(const AResponse:SmallInt;const AAllowedResponses:array of SmallInt): SmallInt
346: Function CheckCrc32( var X : array of Byte; N : Integer; Crc : Cardinal) : Integer;
347: function CheckSynchronize(Timeout: Integer): Boolean
348: Function CheckWin32Version( AMajor : Integer; AMinor : Integer) : Boolean
349: function ChrA(const a: byte): char;
350: Function ClassIDToProgID(const ClassID: TGUID): string;
351: Function ClassNameIs(const Name: string): Boolean
352: Function ClearBit( const Value : Byte; const Bit : TBitRange) : Byte;
353: Function ClearBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
354: Function ClearBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
355: Function ClearBit3( const Value : Word; const Bit : TBitRange) : Word;
356: Function ClearBit4( const Value : Integer; const Bit : TBitRange) : Integer;
```

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

```
442: Function ConvInc( const AValue : Double; const AType, AAmountType : TConvType) : Double;
443: Function ConvInc1(const AValue:Double;const AType:TConvType;const AAmount:Double;const
     AAmountType:TConvType): Double;
444: Function ConvSameValue(const AValue1:Double;const AType1:TConvType;const AValue2:Double;const
     AType2:TConvType):Boolean
445: Function ConvToStr( const AValue : Double; const AType : TConvType) : string
446: Function ConvWithinNext( const AValue, ATest : Double; const AType : TConvType; const AAmount : Double;
     const AAmountType : TConvType) : Boolean
447: Function ConvWithinPrevious(const AValue,ATest:Double;const AType:TConvType; const AAmount:Double;const
     AAmountType: TConvType) : Boolean
448: function Copy(s: AnyString; iFrom, iCount: Longint): AnyString;
449: Function CopyFile( Source, Dest : string; CanOverwrite : Boolean) : Boolean
450: Function CopyFileEx( Source, Dest : string; Flags : FILEOP_FLAGS) : Boolean
451: Function CopyFileTo( const Source, Destination : string) : Boolean
452: function CopyFrom(Source:TStream;Count:Int64):LongInt
453: Function CopyPalette( Palette : HPALETTE) : HPALETTE
454: Function CopyTo( Length : Integer) : string
455: Function CopyTo(stm: IStream; cb: Largeint;out cbRead: Largeint;out cbWritten:Largeint): HResult
456: Function CopyToEOF : string
457: Function CopyToEOL : string
458: Function Cos(e : Extended) : Extended;
459: Function Cosecant( const X : Extended) : Extended
460: Function Cot( const X : Extended) : Extended
461: Function Cotan( const X : Extended) : Extended
462: Function CotH( const X : Extended) : Extended
463: Function Count : Integer
464: Function CountBitsCleared( X : Byte) : Integer;
465: Function CountBitsCleared1( X : Shortint) : Integer;
466: Function CountBitsCleared2( X : Smallint) : Integer;
467: Function CountBitsCleared3( X : Word) : Integer;
468: Function CountBitsCleared4( X : Integer) : Integer;
469: Function CountBitsCleared5( X : Cardinal) : Integer;
470: Function CountBitsCleared6( X : Int64) : Integer;
471: Function CountBitsSet( X : Byte) : Integer;
472: Function CountBitsSet1( X : Word) : Integer;
473: Function CountBitsSet2( X : Smallint) : Integer;
474: Function CountBitsSet3( X : ShortInt) : Integer;
475: Function CountBitsSet4( X : Integer) : Integer;
476: Function CountBitsSet5( X : Cardinal) : Integer;
477: Function CountBitsSet6( X : Int64) : Integer;
478: function CountGenerations(Ancestor,Descendent: TClass): Integer
479: Function Coversine( X : Float) : Float
480: function CRC32(const fileName: string): LongWord;
481: Function CREATEBLOBSTREAM( FIELD : TFIELD; MODE : TBLOBSTREAMMODE) : TSTREAM
482: Function CreateColumns : TDBGridColumns
483: Function CreateDataLink : TGridDataLink
484: Function CreateDir( Dir : string) : Boolean
485: function CreateDir(const Dir: string): Boolean)
486: Function CreateDOSProcessRedirected( const CommandLine, InputFile, OutputFile : string) : Boolean
487: Function CreateEnvironmentBlock(const Options:TEnvironmentOptions;const AdditionalVars:TStrings): PChar
488: Function CREATEFIELD(OWNER:TCOMPONENT;PARENTFIELD:TOBJECTFIELD; const
     FIELDNAME:String;CREATECHILDREN:BOOLEAN):TFIELD
489: Function CreateGlobber( sFilespec : string) : TniRegularExpression
490: Function CreateGrayMappedBmp( Handle : HBITMAP) : HBITMAP
491: Function CreateGrayMappedRes( Instance : THandle; ResName : PChar) : HBITMAP
492: function CreateGUID(out Guid: TGUID): HResult)
493: Function CreateInstance( const unkOuter : IUnknown; const iid : TGUID; out obj) : HResult
494: Function CreateMappedBmp( Handle : HBITMAP; const OldColors, NewColors : array of TColor) : HBITMAP
495: Function CreateMappedRes(Instance:THandle;ResName:PChar;const OldColors,NewColors:array of TColor):HBITMAP
496: Function CreateMessageDialog(const Msg:string; DlgType:TMsgDlgType; Buttons: TMsgDlgButtons) : TForm;
497: Function CreateMessageDialog1(const
     Msg:string;DlgType:TMsgDlgType;Btns:TMsgDlgBtns;DefaultBtn:TMsgDlgBtn):TForm;
498: function CreateOleObject(const ClassName: String): IDispatch;
499: Function CREATEPARAM( FLDTYPE : TFIELDTYPE; const PARAMNAME : String; PARAMTYPE : TPARAMTYPE) : TPARAM
500: Function CreateParameter(const
     Name:WideString;DataType:TDataType;Direction:TParameterDirection;Size:Integer;Value: OleVariant):TParameter
501: Function CreateLocate( DataSet : TDataSet) : TJvLocateObject
502: Function CreateMappedBmp( Handle : HBITMAP; const OldColors, NewColors : array of TColor) : HBITMAP
503: Function CreateMappedRes(Instance:THandle;ResName:PChar;const OldColors,NewColors:array of TColor):HBITMAP
504: Function CreateRecordBuffer( Length : Integer) : TRecordBuffer
505: Function CreateValueBuffer( Length : Integer) : TValueBuffer
506: Function CreatePopupCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode) : TWinControl
507: Function CreateRecordBuffer( Length : Integer) : TRecordBuffer
508: Function CreateRotatedFont( Font : TFont; Angle : Integer) : HFONT
509: Function CreateTwoColorsBrushPattern( Color1, Color2 : TColor) : TBitmap
510: Function CreateValueBuffer( Length : Integer) : TValueBuffer
511: Function CreateHexDump( AOwner : TWinControl) : THexDump
512: Function Csc( const X : Extended) : Extended
513: Function CscH( const X : Extended) : Extended
514: function currencyDecimals: Byte
515: function currencyFormat: Byte
516: function currencyString: String
517: Function CurrentProcessId : TIdPID
518: Function CurrentReadBuffer : string
519: Function CurrentThreadId : TIdPID
520: Function CurrentYear : Word
521: Function CurrToBCD(const Curr: Currency; var BCD: TBcd; Precision: Integer; Decimals: Integer): Boolean
522: Function CurrToStr( Value : Currency) : string;
523: Function CurrToStrF( Value : Currency; FormatSettings : TFormatSettings; Digits: Integer) : string;
```

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

```
612: Function DelChar( const pStr : string; const pChar : Char) : string
613: Function DelEnvironmentVar( const Name : string) : Boolean
614: Function Delete( const MsgNum : Integer) : Boolean
615: Function DeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean) : Boolean
616: Function DeleteFile(const FileName: string): boolean)
617: Function DeleteFileEx( FileName : string; Flags : FILEOP_FLAGS) : Boolean
618: Function DelimiterPosn( const sString : string; const sDelimiters: string): integer;
619: Function DelimiterPosn1(const sString:string;const sDelimiters:string;out cDelimiter: char) : integer;
620: Function DelSpace( const pStr : string) : string
621: Function DelString( const pStr, pDelStr : string) : string
622: Function DelTree( const Path : string) : Boolean
623: Function Depth : Integer
624: Function Description : string
625: Function DescriptionsAvailable : Boolean
626: Function DescriptionToConvFamily( const ADescription : string; out AFamily : TConvFamily) : Boolean
627: Function DescriptionToConvType( const ADescription : string; out AType : TConvType) : Boolean;
628: Function DescriptionToConvType1(const AFamil:TConvFamily;const ADescr:string;out AType:TConvType):Boolean;
629: Function DetectUTF8Encoding( const s : UTF8String) : TEncodeType
630: Function DialogsToPixelsX( const Dialogs : Word) : Word
631: Function DialogsToPixelsY( const Dialogs : Word) : Word
632: Function Digits( const X : Cardinal) : Integer
633: Function DirectoryExists( const Name : string) : Boolean
634: Function DirectoryExists( Directory : string) : Boolean
635: Function DiskFree( Drive : Byte) : Int64
636: function DiskFree(Drive: Byte): Int64)
637: Function DiskInDrive( Drive : Char) : Boolean
638: Function DiskSize( Drive : Byte) : Int64
639: function DiskSize(Drive: Byte): Int64)
640: Function DISPATCHCOMMAND( ACOMMAND : WORD) : BOOLEAN
641: Function DispatchEnabled : Boolean
642: Function DispatchMask : TMask
643: Function DispatchMethodType : TMethodType
644: Function DISPATCHPOPUP( AHANDLE : HMENU) : BOOLEAN
645: Function DispatchRequest( Sender : TObject; Request : TWebRequest; Response : TWebResponse) : Boolean
646: Function DisplayCase( const S : String) : String
647: Function DisplayRect( Code : TDisplayCode) : TRect
648: Function DisplayRect( TextOnly : Boolean) : TRect
649: Function DisplayStream( Stream : TStream) : string
650: TBufferCoord', 'record Char : integer; Line : integer; end
651: TDisplayCoord', 'record Column : integer; Row : integer; end
652: Function DisplayCoord( AColumn, ARow : Integer) : TDisplayCoord
653: Function BufferCoord( AChar, ALine : Integer) : TBufferCoord
654: Function DomainName( const AHost : String) : String
655: Function DosPathToUnixPath( const Path : string) : string
656: Function DottedLineTo( const Canvas : TCanvas; const X, Y : Integer) : Boolean;
657: Function DoubleDecliningBalance( const Cost, Salvage : Extended; Life, Period : Integer) : Extended
658: Function DoubleToBcd( const AValue : Double) : TBcd;
659: Function DoubleToHex( const D : Double) : string
660: Function DoUpdates : Boolean
661: function Dragging: Boolean;
662: Function DrawCaption( p1 : HWND; p2 : HDC; const p3 : TRect; p4 : UINT) : BOOL
663: Function DrawAnimatedRects( hwnd : HWND; idAni : Integer; const lprcFrom, lprcTo : TRect) : BOOL
664: Function DrawEdge( hdc : HDC; var qrc : TRect; edge : UINT; grfFlags : UINT) : BOOL
665: Function DrawFrameControl( DC : HDC; const Rect : TRect; uType, uState : UINT) : BOOL
666: {Works like InputQuery but displays 2edits. If PasswordChar <> #0, the second edit's PasswordChar is set}
667: Function DualInputQuery(const ACaption,Prompt1,Prompt2:string;var AValue1,
     AValue2:string;PasswordChar:Char= #0):Boolean;
668: Function DupeString( const AText : string; ACount : Integer) : string
669: Function Edit : Boolean
670: Function EditCaption : Boolean
671: Function EditText : Boolean
672: Function EditFolderList( Folders : TStrings) : Boolean
673: Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext) : Boolean
674: Function Elapsed( const Update : Boolean) : Cardinal
675: Function EnableProcessPrivilege( const Enable : Boolean; const Privilege : string) : Boolean
676: Function EnableThreadPrivilege( const Enable : Boolean; const Privilege : string) : Boolean
677: Function EncodeDate( Year, Month, Day : Word) : TDateTime
678: function EncodeDate(Year, Month, Day: Word): TDateTime;
679: Function EncodeDateDay( const AYear, ADayOfYear : Word) : TDateTime
680: Function EncodeDateMonthWeek( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word) : TDateTime
681: Function EncodeDateTime(const AYear,AMonth,ADay,AHour,AMinute,ASecond,AMilliSecond: Word): TDateTime
682: Function EncodeDateWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
683: Function EncodeDayOfWeekInMonth( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word) : TDateTime
684: Function EncodeString( s : string) : string
685: Function DecodeString( s : string) : string
686: Function EncodeTime( Hour, Min, Sec, MSec : Word) : TDateTime
687: function EncodeTime(Hour, Min, Sec, MSec: Word): TDateTime;
688: Function EndIP : String
689: Function EndOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
690: Function EndOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
691: Function EndOfAMonth( const AYear, AMonth : Word) : TDateTime
692: Function EndOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
693: Function EndOfAYear( const AYear : Word) : TDateTime
694: Function EndOfTheDay( const AValue : TDateTime) : TDateTime
695: Function EndOfTheMonth( const AValue : TDateTime) : TDateTime
696: Function EndOfTheWeek( const AValue : TDateTime) : TDateTime
697: Function EndOfTheYear( const AValue : TDateTime) : TDateTime
698: Function EndPeriod( const Period : Cardinal) : Boolean
699: Function EndsStr( const ASubText, AText : string) : Boolean
```

```
700: Function EndsText( const ASubText, AText : string) : Boolean
701: Function EnsureMsgIDBrackets( const AMsgID : String) : String
702: Function EnsureRange( const AValue, AMin, AMax : Integer) : Integer;
703: Function EnsureRange1( const AValue, AMin, AMax : Int64) : Int64;
704: Function EnsureRange2( const AValue, AMin, AMax : Double) : Double;
705: Function EOF: boolean
706: Function EOln: boolean
707: Function EqualRect( const R1, R2 : TRect) : Boolean
708: function EqualRect(const R1, R2: TRect): Boolean)
709: Function Equals( Strings : TWideStrings) : Boolean
710: function Equals(Strings: TStrings): Boolean;
711: Function EqualState( oState : TniRegularExpressionState) : boolean
712: Function ErrOutput: Text)
713: function ExceptionParam: String;
714: function ExceptionPos: Cardinal;
715: function ExceptionProc: Cardinal;
716: function ExceptionToString(er: TIFException; Param: String): String;
717: function ExceptionType: TIFException;
718: Function ExcludeTrailingBackslash( S : string) : string
719: function ExcludeTrailingBackslash(const S: string): string)
720: Function ExcludeTrailingPathDelimiter( const APath : string) : string
721: Function ExcludeTrailingPathDelimiter( S : string) : string
722: function ExcludeTrailingPathDelimiter(const S: string): string)
723: function ExecConsoleApp(const AppName, Parameters: String; AppOutput: TStrings): boolean;
724: Function ExecProc : Integer
725: Function ExecSQL : Integer
726: Function ExecSQL( ExecDirect : Boolean) : Integer
727: Function Execute : _Recordset;
728: Function Execute : Boolean
729: Function Execute : Boolean;
730: Function Execute( const SQL : string; Params : TParams; Cache : Boolean; cursor : phDBICur) : Integer
731: Function Execute( const SQL : WideString; Params : TParams; ResultSet : TPSResult) : Integer
732: Function Execute( ParentWnd : HWND) : Boolean
733: Function Execute1(constCommText:WideString;const CType:TCommandType;const ExecuteOptions:TExecuteOptions):
     _Recordset;
734: Function Execute1( const Parameters : OleVariant) : _Recordset;
735: Function Execute1( ParentWnd : HWND) : Boolean;
736: Function Execute2( var RecordsAffected : Integer; const Parameters : OleVariant) : _Recordset;
737: Function ExecuteAction( Action : TBasicAction) : Boolean
738: Function ExecuteDirect( const SQL : WideString) : Integer
739: Function ExecuteFile(const FileName:string;const Params:string;const DefDir:string;ShowCmd:Int):THandle
740: Procedure ExecuteThread2(afunc:TThreadFunction2;thrOK:boolean);AddTypeS('TThreadFunction2','procedure
741: Function CreateThread2(ThreadFunc: TThreadFunction2) : THandle
742: function ExeFileIsRunning(ExeFile: string): boolean;
743: function ExePath: string;
744: function ExePathName: string;
745: Function Exists( AItem : Pointer) : Boolean
746: Function ExitWindows( ExitCode : Cardinal) : Boolean
747: function Exp(x: Extended): Extended;
748: Function ExpandEnvironmentVar( var Value : string) : Boolean
749: Function ExpandFileName( FileName : string) : string
750: function ExpandFileName(const FileName: string): string)
751: Function ExpandUNCFileName( FileName : string) : string
752: function ExpandUNCFileName(const FileName: string): string)
753: Function ExpJ( const X : Float) : Float;
754: Function Exsecans( X : Float) : Float
755: Function Extract( const AByteCount : Integer) : string
756: Function Extract( Item : TClass) : TClass
757: Function Extract( Item : TComponent) : TComponent
758: Function Extract( Item : TObject) : TObject
759: Function ExtractFileDir( FileName : string) : string
760: function ExtractFileDir(const FileName: string): string)
761: Function ExtractFileDrive( FileName : string) : string
762: function ExtractFileDrive(const FileName: string): string)
763: Function ExtractFileExt( FileName : string) : string
764: function ExtractFileExt(const FileName: string): string)
765: Function ExtractFileExtNoDot( const FileName : string) : string
766: Function ExtractFileExtNoDotUpper( const FileName : string) : string
767: Function ExtractFileName( FileName : string) : string
768: function ExtractFileName(const filename: string):string;
769: Function ExtractFilePath( FileName : string) : string
770: function ExtractFilePath(const filename: string):string;
771: Function ExtractRelativePath( BaseName, DestName : string) : string
772: function ExtractRelativePath(const BaseName: string; const DestName: string): string)
773: Function ExtractShortPathName( FileName : string) : string
774: function ExtractShortPathName(const FileName: string): string)
775: function ExtractStrings(Separators, WhiteSpace: TSysCharSet; Content: PChar;Strings: TStrings): Integer
776: function ExtractStrings(Separators:TSysCharSet;WhiteSpace:TSysCharSet;Content:PChar;Str:TStrings): Integer)
777: Function Fact(numb: integer): Extended;
778: Function FactInt(numb: integer): int64;
779: Function Factorial( const N : Integer) : Extended
780: Function FahrenheitToCelsius( const AValue : Double) : Double
781: function FalseBoolStrs: array of string
782: Function Fetch(var AInput:string;const ADelim:string;const ADelete:Bool;const ACaseSensitive:Bool):string
783: Function FetchCaseInsensitive(var AInput:string; const ADelim:string; const ADelete:Boolean): string
784: Function Fibo(numb: integer): Extended;
785: Function FiboInt(numb: integer): Int64;
786: Function Fibonacci( const N : Integer) : Integer
787: Function FIELDBYNAME( const FIELDNAME : STRING) : TFIELD
```

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

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

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

```
1049: function GetCurrentDir: string)
1050: Function GetCurrentFolder : string
1051: Function GETCURRENTRECORD( BUFFER : PCHAR) : BOOLEAN
1052: Function GetCurrentProcessId : TIdPID
1053: Function GetCurrentThreadHandle : THandle
1054: Function GetCurrentThreadID: LongWord; stdcall;
1055: Function GetCustomHeader( const Name : string) : String
1056: Function GetDataItem( Value : Pointer) : Longint
1057: Function GetDataLinkFiles( FileNames : TWideStrings; Directory : string) : Integer;
1058: Function GetDataLinkFiles1( FileNames : TStrings; Directory : string) : Integer;
1059: Function GETDATASIZE : INTEGER
1060: Function GetDC(hdwnd: HWND): HDC;
1061: Function GetDefaultFileExt( const MIMEType : string) : string
1062: Function GetDefaults : Boolean
1063: Function GetDefaultSchemaName : WideString
1064: Function GetDefaultStreamLoader : IStreamLoader
1065: Function GetDesktopDirectoryFolder : string
1066: Function GetDesktopFolder : string
1067: Function GetDFAState( oStates : TList) : TniRegularExpressionState
1068: Function GetDirectorySize( const Path : string) : Int64
1069: Function GetDisplayWidth : Integer
1070: Function GetDLLVersion( const DLLName : string; var pdwMajor, pdwMinor : Integer) : Boolean
1071: Function GetDomainName : string
1072: Function GetDriverRegistryFile( DesignMode : Boolean) : string
1073: function GetDriveType(rootpath: pchar): cardinal;
1074: Function GetDriveTypeStr( const Drive : Char) : string
1075: Function GetEnumerator : TFavoriteLinkItemsEnumerator
1076: Function GetEnumerator : TListItemsEnumerator
1077: Function GetEnumerator : TTaskDialogButtonsEnumerator
1078: Function GetEnumerator : TToolBarEnumerator
1079: Function GetEnumerator : TTreeNodesEnumerator
1080: Function GetEnumerator : TWideStringsEnumerator
1081: Function GetEnvVar( const VarName : string) : string
1082: Function GetEnvironmentVar( const AVariableName : string) : string
1083: Function GetEnvironmentVariable( const VarName : string) : string
1084: Function GetEnvironmentVar( const Name : string; var Value : string; Expand : Boolean) : Boolean
1085: Function GetEnvironmentVars( const Vars : TStrings; Expand : Boolean) : Boolean
1086: Function getEnvironmentString: string;
1087: Function GetExceptionHandler : TObject
1088: Function GetFavoritesFolder : string
1089: Function GetFieldByName( const Name : string) : string
1090: Function GetFieldInfo( const Origin : Widestring; var FieldInfo : TFieldInfo) : Boolean
1091: Function GetFieldValue( ACol : Integer) : string
1092: Function GetFileAgeCoherence( const FileName : string) : Boolean
1093: Function GetFileCreation( const FileName : string) : TFileTime
1094: Function GetFileCreationTime( const Filename : string) : TDateTime
1095: Function GetFileInformation( const FileName : string) : TSearchRec
1096: Function GetFileLastAccess( const FileName : string) : TFileTime
1097: Function GetFileLastWrite( const FileName : string) : TFileTime
1098: Function GetFileList(FileList: TStringlist; apath: string): TStringlist;
1099: Function GetFileList1(apath: string): TStringlist;
1100: Function GetFileMIMEType( const AFileName : string) : string
1101: Function GetFileSize( const FileName : string) : Int64
1102: Function GetFileVersion( AFileName : string) : Cardinal
1103: Function GetFileVersion( const AFilename : string) : Cardinal
1104: Function GetFileSize2(Handle: Integer; x: Integer): Integer; stdcall;
1105: Function GetFileDate(aFile:string; aWithTime:Boolean):string;
1106: Function GetFilterData( Root : PExprNode) : TExprData
1107: Function getFirstChild : LongInt
1108: Function getFirstChild : TTreeNode
1109: Function GetFirstDelimitedToken( const cDelim : char; const cStr : string) : string
1110: Function GetFirstNode : TTreeNode
1111: Function GetFontsFolder : string
1112: Function GetFormulaValue( const Formula : string) : Extended
1113: Function GetFreePageFileMemory : Integer
1114: Function GetFreePhysicalMemory : Integer
1115: Function GetFreeSystemResources( const ResourceType : TFreeSysResKind) : Integer;
1116: Function GetFreeSystemResources1 : TFreeSystemResources;
1117: Function GetFreeVirtualMemory : Integer
1118: Function GetFromClipboard : Boolean
1119: Function GetFullURI( const AOptionalFileds : TIdURIOptionalFieldsSet) : String
1120: Function GetGBitmap( Value : TBitmap) : TBitmap
1121: Function GetGMTDateByName( const AFileName : TIdFileName) : TDateTime
1122: Function GetGroupState( Level : Integer) : TGroupPosInds
1123: Function GetHandle : HWND
1124: Function GETHELPCONTEXT( VALUE : INTEGER; BYCOMMAND : BOOLEAN) : THELPCONTEXT
1125: function GetHexArray(ahexdig: THexArray): THexArray;
1126: Function GetHighLightColor( const Color : TColor; Luminance : Integer) : TColor
1127: function GetHINSTANCE: longword;
1128: Function GetHistoryFolder : string
1129: Function GetHitTestInfoAt( X, Y : Integer) : THitTests
1130: function getHMODULE: longword;
1131: Function GetHostByName(const AComputerName: String): String;
1132: Function GetHostName : string
1133: Function getHostIP: string;
1134: Function GetHotSpot : TPoint
1135: Function GetHueBitmap( Value : TBitmap) : TBitmap
1136: Function GetImageBitmap : HBITMAP
1137: Function GETIMAGELIST : TCUSTOMIMAGELIST
```

```
1138: Function GetIncome( const aNetto : Currency) : Currency
1139: Function GetIncome( const aNetto : Extended) : Extended
1140: Function GetIncome( const aNetto : Extended): Extended
1141: Function GetIncome(const aNetto : Extended) : Extended
1142: function GetIncome(const aNetto: Currency): Currency
1143: Function GetIncome2( const aNetto : Currency) : Currency
1144: Function GetIncome2( const aNetto : Currency): Currency
1145: Function getIndex_Attrs( tag : string; var idx : Integer; var Attrs : string) : string
1146: Function GETINDEXFORFIELDS( const FIELDS : String; CASEINSENSITIVE : BOOLEAN) : TINDEXDEF
1147: Function GetIndexForOrderBy( const SQL : WideString; DataSet : TDataSet) : TIndexDef
1148: Function GetInstRes(Instance:THandle;ResType:TResType;const
      Name:string;Width:Integer;LoadFlags:TLoadResources;MaskColor:TColor):Boolean;
1149: Function
      GetInstRes1(Instance:THandle;ResType:TResType;ResID:DWORD;Width:Integer;LoadFlags:TLoadResources;MaskColor:
      TColor):Boolean;
1150: Function GetIntelCacheDescription( const D : Byte) : string
1151: Function GetInteractiveUserName : string
1152: Function GetInternetCacheFolder : string
1153: Function GetInternetFormattedFileTimeStamp( const AFilename : String) : String
1154: Function GetIPAddress( const HostName : string) : string
1155: Function GetIP( const HostName : string) : string
1156: Function GetIPHostByName(const AComputerName: String): String;
1157: Function GetIsAdmin: Boolean;
1158: Function GetItem( X, Y : Integer) : LongInt
1159: Function GetItemAt( X, Y : Integer) : TListItem
1160: Function GetItemHeight(Font: TFont): Integer;
1161: Function GetItemPath( Index : Integer) : string
1162: Function GetKeyFieldNames( List : TStrings) : Integer;
1163: Function GetKeyFieldNames1( List : TWideStrings) : Integer;
1164: Function GetKeyState( const VirtualKey : Cardinal) : Boolean
1165: Function GetLastChild : LongInt
1166: Function GetLastChild : TTreeNode
1167: Function GetLastDelimitedToken( const cDelim : char; const cStr : string) : string
1168: function GetLastError: Integer
1169: Function GetLAT_CONV_FACTOR: double;  //for WGS84 power(1 - 1 / 298.257223563, 2);
1170: Function GetLoader( Ext : string) : TBitmapLoader
1171: Function GetLoadFilter : string
1172: Function GetLocalComputerName : string
1173: Function GetLocaleChar( Locale, LocaleType : Integer; Default : Char) : Char
1174: Function GetLocaleStr( Locale, LocaleType : Integer; Default : string) : string
1175: Function GetLocalUserName : string
1176: Function GetLoginUsername : WideString
1177: function getLongDayNames: string)
1178: Function GetLongHint(const hint: string): string
1179: function getLongMonthNames: string)
1180: Function GetMacAddresses( const Machine : string; const Addresses : TStrings) : Integer
1181: Function GetMainAppWndFromPid( PID : DWORD) : HWND
1182: Function GetMaskBitmap : HBITMAP
1183: Function GetMaxAppAddress : Integer
1184: Function GetMciErrorMessage( const MciErrNo : MCIERROR) : string
1185: Function GetMemoryLoad : Byte
1186: Function GetMIMEDefaultFileExt( const MIMEType : string) : TIdFileName
1187: Function GetMIMETypeFromFile( const AFile : string) : string
1188: Function GetMIMETypeFromFile( const AFile : TIdFileName) : string
1189: Function GetMinAppAddress : Integer
1190: Function GetModule : TComponent
1191: Function GetModuleHandle( ModuleName : PChar) : HMODULE
1192: Function GetModuleName( Module : HMODULE) : string
1193: Function GetModulePath( const Module : HMODULE) : string
1194: Function GetModuleFileName(Module: Integer; Filename: PChar;Size: Integer): Integer; stdcall;
1195: Function GetCommandLine: PChar; stdcall;
1196: Function GetMonochromeBitmap( Value : TBitmap) : TBitmap
1197: Function GetMultiN(aval: integer): string;
1198: Function GetName : String
1199: Function GetNearestItem( Point : TPoint; Direction : TSearchDirection) : TListItem
1200: Function GetNethoodFolder : string
1201: Function GetNext : TTreeNode
1202: Function GetNextChild( Value : LongInt) : LongInt
1203: Function GetNextChild( Value : TTreeNode) : TTreeNode
1204: Function GetNextDelimitedToken( const cDelim : char; var cStr : String) : String
1205: Function GetNextItem( StartItem: TListItem;Direction:TSearchDirection;States:TItemStates) : TListItem
1206: Function GetNextPacket : Integer
1207: Function getNextSibling : TTreeNode
1208: Function GetNextVisible : TTreeNode
1209: Function GetNode( ItemId : HTreeItem) : TTreeNode
1210: Function GetNodeAt( X, Y : Integer) : TTreeNode
1211: Function GetNodeDisplayWidth( Node : TOutlineNode) : Integer
1212: function GetNumberOfProcessors: longint;
1213: Function GetNumLockKeyState : Boolean
1214: Function GetObjectProperty( Instance : TPersistent; const PropName : string) : TObject
1215: Function GetOnlyTransitionOn( cChar : char) : TniRegularExpressionState
1216: Function GetOptionalParam( const ParamName : string) : OleVariant
1217: Function GetOSName: string;
1218: Function GetOSVersion: string;
1219: Function GetOSNumber: string;
1220: Function GetOsVersionInfo: TOSVersionInfo;      //thx to wischnewski
1221: Function GetPackageModuleHandle( PackageName : PChar) : HMODULE
1222: function GetPageSize: Cardinal;
1223: Function GetParameterFileName : string
```

```
1224: Function GetParams( var OwnerData : OleVariant) : OleVariant
1225: Function GETPARENTCOMPONENT : TCOMPONENT
1226: Function GetParentForm(control: TControl): TForm
1227: Function GETPARENTMENU : TMENU
1228: Function GetPassword : Boolean
1229: Function GetPassword : string
1230: Function GetPersonalFolder : string
1231: Function GetPidFromProcessName( const ProcessName : string) : DWORD
1232: Function GetPosition : TPoint
1233: Function GetPrev : TTreeNode
1234: Function GetPrevChild( Value : LongInt) : LongInt
1235: Function GetPrevChild( Value : TTreeNode) : TTreeNode
1236: Function getPrevSibling : TTreeNode
1237: Function GetPrevVisible : TTreeNode
1238: Function GetPrinthoodFolder : string
1239: Function GetPrivilegeDisplayName( const PrivilegeName : string) : string
1240: Function getProcessList: TStrings;
1241: Function GetProcessId : TIdPID
1242: Function GetProcessNameFromPid( PID : DWORD) : string
1243: Function GetProcessNameFromWnd( Wnd : HWND) : string
1244: Function GetProcessMemoryInfo(Process: THandle;ppsmemCounters: TProcessMemoryCounters;cb: DWORD):BOOL
1245: Function getProcessAllMemory(ProcessID : DWORD): TProcessMemoryCounters;
1246: Function getProcessMemoryInfo2(ProcessID : DWORD): TProcessMemoryCounters;
1247: Function GetProgramFilesFolder : string
1248: Function GetProgramsFolder : string
1249: Function GetProxy : string
1250: Function GetQuoteChar : WideString
1251: Function GetQrCode4(Width,Height:Word; Correct_Level:string;const
      Data:string;aformat:string):TLinearBitmap;
1252: Function GetQrCodetoFile(Width,Height:Word;Correct_Level:string;const
      Data:string;aformat:string):TLinearBitmap;
1253: Function GetRate : Double
1254: Function getPerfTime: string;
1255: Function getRuntime: string;
1256: Function GetRBitmap( Value : TBitmap) : TBitmap
1257: Function GetReadableName( const AName : string) : string
1258: Function GetRecentDocs : TStringList
1259: Function GetRecentFolder : string
1260: Function GetRecords( Count : Integer; out RecsOut : Integer; Options : Integer) : OleVariant;
1261: Function GetRecords1(Count:Integer; out RecsOut:Integer;Options:Integer;const CommandText:WideString;var
      Params, OwnerData : OleVariant) : OleVariant;
1262: Function GetRecordset( const CommandText : WideString; ConnectionString : WideString) : _Recordset
1263: Function GetRegisteredCompany : string
1264: Function GetRegisteredOwner : string
1265: Function GetResource(ResType:TResType;const
      Name:string;Width:Integer;LoadFlags:TLoadResources;MaskColor:TColor:Boolean
1266: Function GetResourceName( ObjStream : TStream; var AName : string) : Boolean
1267: Function GetResponse( const AAllowedResponses : array of SmallInt) : SmallInt;
1268: Function GetResponse1( const AAllowedResponse : SmallInt) : SmallInt;
1269: Function GetRValue( rgb : DWORD) : Byte
1270: Function GetGValue( rgb : DWORD) : Byte
1271: Function GetBValue( rgb : DWORD) : Byte
1272: Function GetCValue( cmyk : COLORREF) : Byte
1273: Function GetMValue( cmyk : COLORREF) : Byte
1274: Function GetYValue( cmyk : COLORREF) : Byte
1275: Function GetKValue( cmyk : COLORREF) : Byte
1276: Function CMYK( c, m, y, k : Byte) : COLORREF
1277: Function GetOSName: string;
1278: Function GetProcAddress( hModule : HMODULE; lpProcName : LPCSTR) : FARPROC
1279: Function GetProcAddress(Module : HMODULE; Proc : PChar): Dword
1280: Function GetSafeCallExceptionMsg : String
1281: Function GetSaturationBitmap( Value : TBitmap) : TBitmap
1282: Function GetSaveFilter : string
1283: Function GetSaver( Ext : string) : TBitmapLoader
1284: Function GetScrollLockKeyState : Boolean
1285: Function GetSearchString : string
1286: Function GetSelections( AList : TList) : TTreeNode
1287: function GETSELTEXTBUF(BUFFER:PCHAR;BUFSIZE:INTEGER):INTEGER
1288: Function GetSendToFolder : string
1289: Function GetServer : IAppServer
1290: Function GetServerList : OleVariant
1291: Function GetShadowColor( const Color : TColor; Luminance : Integer) : TColor
1292: Function GetShellProcessHandle : THandle
1293: Function GetShellProcessName : string
1294: Function GetShellVersion : Cardinal
1295: function getShortDayNames: string)
1296: Function GetShortHint(const hint: string): string
1297: function getShortMonthNames: string)
1298: Function GetSizeOfFile( const FileName : string) : Int64;
1299: Function GetSizeOfFile1( Handle : THandle) : Int64;
1300: Function GetStdHandle(nStdHandle: Integer): Integer; stdcall;
1301: Function GetStartmenuFolder : string
1302: Function GetStartupFolder : string
1303: Function GetStringProperty( Instance : TPersistent; const PropName : string) : WideString
1304: Function GetSuccessor( cChar : char) : TniRegularExpressionState
1305: Function GetSwapFileSize : Integer
1306: Function GetSwapFileUsage : Integer
1307: Function GetSystemLocale : TIdCharSet
1308: Function GetSystemMetrics( nIndex : Integer) : Integer
```

```
1309: Function GetSystemPathSH(Folder: Integer): TFilename ;
1310: Function GetTableNameFromQuery( const SQL : Widestring) : Widestring
1311: Function GetTableNameFromSQL( const SQL : WideString) : WideString
1312: Function GetTableNameFromSQLEx( const SQL : WideString; IdOption : IDENTIFIEROption) : WideString
1313: Function GetTasksList( const List : TStrings) : Boolean
1314: Function getTeamViewerID: string;
1315: Function GetTemplatesFolder : string
1316: Function GetText : PwideChar
1317: function GetText:PChar
1318: Function GetTextBuf( Buffer : PChar; BufSize : Integer) : Integer
1319: function GETTEXTBUF(BUFFER:PCHAR;BUFSIZE:INTEGER):INTEGER
1320: Function GetTextItem( const Value : string) : Longint
1321: function GETTEXTLEN:INTEGER
1322: Function GetThreadLocale: Longint; stdcall
1323: Function GetCurrentThreadID: LongWord; stdcall;
1324: Function GetTickCount : Cardinal
1325: Function GetTickDiff( const AOldTickCount, ANewTickCount : Cardinal) : Cardinal
1326: Function GetTicketNr : longint
1327: Function GetTime : Cardinal
1328: Function GetTime : TDateTime
1329: Function GetTimeout : Integer
1330: Function GetTimeStr: String
1331: Function GetTimeString: String
1332: Function GetTodayFiles(startdir, amask: string): TStringlist;
1333: Function getTokenCounts : integer
1334: Function GetTotalPageFileMemory : Integer
1335: Function GetTotalPhysicalMemory : Integer
1336: Function GetTotalVirtualMemory : Integer
1337: Function GetUniqueFileName( const APath, APrefix, AExt : String) : String
1338: Function GetUseNowForDate : Boolean
1339: Function GetUserDomainName( const CurUser : string) : string
1340: Function GetUserName : string
1341: Function GetUserName: string;
1342: Function GetUserObjectName( hUserObject : THandle) : string
1343: Function GetValueBitmap( Value : TBitmap) : TBitmap
1344: Function GetValueMSec : Cardinal
1345: Function GetValueStr : String
1346: Function GetVersion: int;
1347: Function GetVersionString(FileName: string): string;
1348: Function GetVisibleNode( Index : LongInt) : TOutlineNode
1349: Function GetVolumeFileSystem( const Drive : string) : string
1350: Function GetVolumeName( const Drive : string) : string
1351: Function GetVolumeSerialNumber( const Drive : string) : string
1352: Function GetWebAppServices : IWebAppServices
1353: Function GetWebRequestHandler : IWebRequestHandler
1354: Function GetWindowCaption( Wnd : HWND) : string
1355: Function GetWindowDC(hdwnd: HWND): HDC;
1356: Function GetWindowIcon( Wnd : HWND; LargeIcon : Boolean) : HICON
1357: Function GetWindowRect(hwnd: HWND; arect: TRect): Boolean
1358: Function GetWindowsComputerID : string
1359: function GetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
1360: Function GetWindowsFolder : string
1361: Function GetWindowsServicePackVersion : Integer
1362: Function GetWindowsServicePackVersionString : string
1363: Function GetWindowsSystemFolder : string
1364: Function GetWindowsTempFolder : string
1365: Function GetWindowsUserID : string
1366: Function GetWindowsVersion : TWindowsVersion
1367: Function GetWindowsVersionString : string
1368: Function GmtOffsetStrToDateTime( S : string) : TDateTime
1369: Function GMTToLocalDateTime( S : string) : TDateTime
1370: Function GotoKey : Boolean
1371: Function GradToCycle( const Grads : Extended) : Extended
1372: Function GradToDeg( const Grads : Extended) : Extended
1373: Function GradToDeg( const Value : Extended) : Extended;
1374: Function GradToDeg1( const Value : Double) : Double;
1375: Function GradToDeg2( const Value : Single) : Single;
1376: Function GradToRad( const Grads : Extended) : Extended
1377: Function GradToRad( const Value : Extended) : Extended;
1378: Function GradToRad1( const Value : Double) : Double;
1379: Function GradToRad2( const Value : Single) : Single;
1380: Function Gray32( const Intensity : Byte; const Alpha : Byte) : TColor32
1381: Function GreenComponent( const Color32 : TColor32) : Integer
1382: function GUIDToString(const GUID: TGUID): string)
1383: Function HandleAllocated : Boolean
1384: function HandleAllocated: Boolean;
1385: Function HandleRequest : Boolean
1386: Function HandleRequest( Request : TWebRequest; Response : TWebResponse) : Boolean
1387: Function HarmonicMean( const X : TDynFloatArray) : Float
1388: Function HasAsParent( Value : TTreeNode) : Boolean
1389: Function HASCHILDDEFS : BOOLEAN
1390: Function HasCurValues : Boolean
1391: Function HasExtendCharacter( const s : UTF8String) : Boolean
1392: Function HasFormat( Format : Word) : Boolean
1393: Function HashValue( AStream : TStream) : T5x4LongWordRecord;
1394: Function HashValue(AStream : TStream) : T4x4LongWordRecord
1395: Function HashValue(AStream: TStream): LongWord
1396: Function HashValue(AStream: TStream): Word
1397: Function HashValue1( AStream : TStream; const ABeginPos, AEndPos : Int64) : T5x4LongWordRecord;
```

```
1398: Function HashValue1(AStream : TStream): T4x4LongWordRecord
1399: Function HashValue128(const ASrc: string): T4x4LongWordRecord;
1400: Function HashValue128Stream(AStream: TStream): T4x4LongWordRecord;
1401: Function HashValue16( const ASrc : string) : Word;
1402: Function HashValue16stream( AStream : TStream) : Word;
1403: Function HashValue32( const ASrc : string) : LongWord;
1404: Function HashValue32Stream( AStream : TStream) : LongWord;
1405: Function HasMergeConflicts : Boolean
1406: Function hasMoreTokens : boolean
1407: Function HASPARENT : BOOLEAN
1408: function HasParent: Boolean
1409: Function HasTransaction( Transaction : TDBXTransaction) : Boolean
1410: Function HasUTF8BOM( S : TStream) : boolean;
1411: Function HasUTF8BOM1( S : AnsiString) : boolean;
1412: Function Haversine( X : Float) : Float
1413: Function Head( s : string; const subs : string; var tail : string) : string
1414: function HELPCOMMAND(COMMAND:INTEGER;DATA:LONGINT):BOOLEAN
1415: function HELPCONTEXT(CONTEXT:THELPCONTEXT):BOOLEAN
1416: function HELPJUMP(JUMPID:STRING):BOOLEAN
1417: Function HeronianMean( const a, b : Float) : Float
1418: function HexStrToStr(Value: string): string;
1419: function HexToBin(Text,Buffer:PChar; BufSize:Integer):Integer;
1420: function HexToBin2(HexNum: string): string;
1421: Function HexToDouble( const Hex : string) : Double
1422: function HexToInt(hexnum: string): LongInt;
1423: function HexToStr(Value: string): string;
1424: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
1425: function Hi(vdat: word): byte;
1426: function HiByte(W: Word): Byte)
1427: function High: Int64;
1428: Function HighlightCell(DataCol,DataRow: Integer; const Value:string; AState:TGridDrawState): Boolean
1429: function HINSTANCE: longword;
1430: function HiWord(l: DWORD): Word)
1431: function HMODULE: longword;
1432: Function HourOf( const AValue : TDateTime) : Word
1433: Function HourOfTheDay( const AValue : TDateTime) : Word
1434: Function HourOfTheMonth( const AValue : TDateTime) : Word
1435: Function HourOfTheWeek( const AValue : TDateTime) : Word
1436: Function HourOfTheYear( const AValue : TDateTime) : Word
1437: Function HoursBetween( const ANow, AThen : TDateTime) : Int64
1438: Function HourSpan( const ANow, AThen : TDateTime) : Double
1439: Function HSLToRGB1( const H, S, L : Single) : TColor32;
1440: Function HTMLDecode( const AStr : String) : String
1441: Function HTMLEncode( const AStr : String) : String
1442: Function HTMLEscape( const Str : string) : string
1443: Function HtmlTable( DataSet : TDataSet; DataSetHandler : TDSTableProducer; MaxRows : Integer) : string
1444: Function HTTPDecode( const AStr : String) : string
1445: Function HTTPEncode( const AStr : String) : string
1446: Function Hypot( const X, Y : Extended) : Extended
1447: Function IBMax( n1, n2 : Integer) : Integer
1448: Function IBMin( n1, n2 : Integer) : Integer
1449: Function IBRandomString( iLength : Integer) : String
1450: Function IBRandomInteger( iLow, iHigh : Integer) : Integer
1451: Function IBStripString( st : String; CharsToStrip : String) : String
1452: Function IBFormatIdentifier( Dialect : Integer; Value : String) : String
1453: Function IBFormatIdentifierValue( Dialect : Integer; Value : String) : String
1454: Function IBExtractIdentifier( Dialect : Integer; Value : String) : String
1455: Function IBQuoteIdentifier( Dialect : Integer; Value : String) : String
1456: Function IBAddIBParamSQLForDetail(Params:TParams;SQL:string;Native:Boolean;Dialect:Integer):string
1457: Procedure IBDecomposeDatabaseName(DatabaseName:String;var ServerName,Protocol,DatabasePath:String)
1458: Function IconToBitmap( Ico : HICON) : TBitmap
1459: Function IconToBitmap2( Ico : HICON; Size : Integer; TransparentColor : TColor) : TBitmap
1460: Function IconToBitmap3( Ico : HICON; Size : Integer; TransparentColor : TColor) : TBitmap
1461: function IdentToCharset(const Ident: string; var Charset: Longint): Boolean)
1462: function IdentToColor(const Ident: string; var Color: Longint): Boolean)
1463: function IdentToCursor(const Ident: string; var cursor: Longint): Boolean;
1464: Function IdGetDefaultCharSet : TIdCharSet
1465: function IDispatchInvoke(Self:IDispatch;ProperSet:Boolean;const Name:String;Par:array of variant):variant
1466: Function IdPorts2 : TStringList
1467: Function IdToMib( const Id : string) : string
1468: Function IdSHA1Hash(apath: string): string;
1469: Function IdHashSHA1(apath: string): string;
1470: Function IfStr( const bCondition : boolean; const sTrue : string; const sFalse : string) : string
1471: Function IfThen( AValue : Boolean; const ATrue : string; AFalse : string) : string;
1472: Function iif1( ATest : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer;
1473: Function iif2( ATest : Boolean; const ATrue : string; const AFalse : string) : string;
1474: Function iif3( ATest : Boolean; const ATrue : Boolean; const AFalse : Boolean) : Boolean;
1475: function ImportTest(S1: string;s2:longint; s3:Byte; s4:word; var s5:string): string;
1476: Function IncDay( const AValue : TDateTime; const ANumberOfDays : Integer) : TDateTime
1477: Function IncHour( const AValue : TDateTime; const ANumberOfHours : Int64) : TDateTime
1478: Function IncLimit( var B : Byte; const Limit : Byte; const Incr : Byte) : Byte;
1479: Function IncLimit1( var B : Shortint; const Limit : Shortint; const Incr : Shortint) : Shortint;
1480: Function IncLimit2( var B : Smallint; const Limit : Smallint; const Incr : Smallint) : Smallint;
1481: Function IncLimit3( var B : Word; const Limit : Word; const Incr : Word) : Word;
1482: Function IncLimit4( var B : Integer; const Limit : Integer; const Incr : Integer) : Integer;
1483: Function IncLimit5( var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal) : Cardinal;
1484: Function IncLimit6( var B : Int64; const Limit : Int64; const Incr : Int64) : Int64;
1485: Function IncLimitClamp( var B : Byte; const Limit : Byte; const Incr : Byte) : Byte;
1486: Function IncLimitClamp1( var B : Shortint; const Limit : Shortint; const Incr : Shortint) : Shortint;
```

```
1487: Function IncLimitClamp2( var B : Smallint; const Limit : Smallint; const Incr : Smallint) : Smallint;
1488: Function IncLimitClamp3( var B : Word; const Limit : Word; const Incr : Word) : Word;
1489: Function IncLimitClamp4( var B : Integer; const Limit : Integer; const Incr : Integer) : Integer;
1490: Function IncLimitClamp5( var B : Cardinal; const Limit : Cardinal; const Incr : Cardinal) : Cardinal;
1491: Function IncLimitClamp6( var B : Int64; const Limit : Int64; const Incr : Int64) : Int64;
1492: Function IncludeTrailingBackslash( S : string) : string
1493: function IncludeTrailingBackslash(const S: string): string)
1494: Function IncludeTrailingPathDelimiter( const APath : string) : string
1495: Function IncludeTrailingPathDelimiter( S : string) : string
1496: function IncludeTrailingPathDelimiter(const S: string): string)
1497: Function IncludeTrailingSlash( const APath : string) : string
1498: Function IncMilliSecond( const AValue : TDateTime; const ANumberOfMilliSeconds : Int64) : TDateTime
1499: Function IncMinute( const AValue : TDateTime; const ANumberOfMinutes : Int64) : TDateTime
1500: Function IncMonth( DateTime : TDateTime; NumberOfMonths : Integer) : TDateTime
1501: function IncMonth(const DateTime: TDateTime; NumberOfMonths: Integer): TDateTime)
1502: Function IncSecond( const AValue : TDateTime; const ANumberOfSeconds : Int64) : TDateTime
1503: Function IncWeek( const AValue : TDateTime; const ANumberOfWeeks : Integer) : TDateTime
1504: Function IncYear( const AValue : TDateTime; const ANumberOfYears : Integer) : TDateTime
1505: Function IndexOf( AClass : TClass) : Integer
1506: Function IndexOf( AComponent : TComponent) : Integer
1507: Function IndexOf( AObject : TObject) : Integer
1508: Function INDEXOF( const ANAME : String) : INTEGER
1509: Function IndexOf( const DisplayName : string) : Integer
1510: Function IndexOf( const Item : TBookmarkStr) : Integer
1511: Function IndexOf( const S : WideString) : Integer
1512: Function IndexOf( const View : TJclFileMappingView) : Integer
1513: Function INDEXOF( FIELD : TFIELD) : INTEGER
1514: Function IndexOf( ID : LCID) : Integer
1515: Function INDEXOF( ITEM : TMENUITEM) : INTEGER
1516: Function IndexOf( Value : TListItem) : Integer
1517: Function IndexOf( Value : TTreeNode) : Integer
1518: function IndexOf(const S: string): Integer;
1519: Function IndexOfName( const Name : WideString) : Integer
1520: function IndexOfName(Name: string): Integer;
1521: Function IndexOfObject( AObject : TObject) : Integer
1522: function IndexofObject(AObject:tObject):Integer
1523: Function IndexOfTabAt( X, Y : Integer) : Integer
1524: Function IndexStr( const AText : string; const AValues : array of string) : Integer
1525: Function IndexText( const AText : string; const AValues : array of string) : Integer
1526: Function IndexOfInteger( AList : TStringList; Value : Variant) : Integer
1527: Function IndexOfFloat( AList : TStringList; Value : Variant) : Integer
1528: Function IndexOfDate( AList : TStringList; Value : Variant) : Integer
1529: Function IndexOfString( AList : TStringList; Value : Variant) : Integer
1530: Function IndyCompareStr( const A1 : string; const A2 : string) : Integer
1531: Function IndyGetHostName : string
1532: Function IndyInterlockedDecrement( var I : Integer) : Integer
1533: Function IndyInterlockedExchange( var A : Integer; B : Integer) : Integer
1534: Function IndyInterlockedExchangeAdd( var A : Integer; B : Integer) : Integer
1535: Function IndyInterlockedIncrement( var I : Integer) : Integer
1536: Function IndyLowerCase( const A1 : string) : string
1537: Function IndyStrToBool( const AString : String) : Boolean
1538: Function IndyUpperCase( const A1 : string) : string
1539: Function InitCommonControl( CC : Integer) : Boolean
1540: Function InitTempPath : string
1541: Function InMainThread : boolean
1542: Function inOpArray( W : WideChar; sets : array of WideChar) : boolean
1543: Function Input: Text)
1544: Function InputBox( const ACaption, APrompt, ADefault : string) : string
1545: Function InputBox(const ACaption: string; const APrompt: string; const ADefault: string): string)
1546: Function InputLn(const AMask: string; AEcho:Boolean;ATabWidth:Integer;AMaxLineLength:Integer): string
1547: Function InputQuery( const ACaption, APrompt : string; var Value : string) : Boolean
1548: function InputQuery(const ACaption: string; const APrompt: string; var Value: string): Boolean)
1549: Function InquireSignal( RtlSigNum : Integer) : TSignalState
1550: Function InRangeR( const A, Min, Max : Double) : Boolean
1551: function Insert( Index : Integer) : TCollectionItem
1552: Function Insert( Index : Integer) : TComboExItem
1553: Function Insert( Index : Integer) : THeaderSection
1554: Function Insert( Index : Integer) : TListItem
1555: Function Insert( Index : Integer) : TStatusPanel
1556: Function Insert( Index : Integer) : TWorkArea
1557: Function Insert( Index : LongInt; const Text : string) : LongInt
1558: Function Insert( Sibling : TTreeNode; const S : string) : TTreeNode
1559: Function INSERTNEWLINEAFTER( AITEM : TMENUITEM) : INTEGER
1560: Function INSERTNEWLINEBEFORE( AITEM : TMENUITEM) : INTEGER
1561: Function InsertNode( Node, Sibling : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
1562: Function InsertObject( Index : LongInt; const Text : string; const Data : Pointer) : LongInt
1563: Function InsertObject( Sibling : TTreeNode; const S : string; Ptr : Pointer) : TTreeNode
1564: Function Instance : Longint
1565: function InstanceSize: Longint
1566: Function Int(e : Extended) : Extended;
1567: function Int64ToStr(i: Int64): String;
1568: Function IntegerToBcd( const AValue : Integer) : TBcd
1569: Function Intensity( const Color32 : TColor32) : Integer;
1570: Function Intensity( const R, G, B : Single) : Single;
1571: Function InterestPayment(const Rate:Extended;Period,NPeriods:Integer;const PresentValue,
      FutureValue:Extended; PaymentTime : TPaymentTime) : Extended
1572: Function InterestRate(NPeriods:Integer;const Payment,PresentVal,
      FutureVal:Extended;PaymentTime:TPaymentTime): Extended
1573: Function InternalDecodeDate( DateTime : TDateTime; var Year, Month, Day, DOW : Word) : Boolean
```

```
1574: Function InternalRateOfReturn( const Guess : Extended; const CashFlows : array of Double) : Extended
1575: Function InternalUpdateRecord( Tree : TUpdateTree) : Boolean
1576: Function IntersectRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
1577: function IntersectRect(out Rect: TRect; const R1, R2: TRect): Boolean)
1578: Function IntMibToStr( const Value : string) : string
1579: Function IntPower( const Base : Extended; const Exponent : Integer) : Extended
1580: Function IntToBin( Value : cardinal) : string
1581: Function IntToHex( Value : Integer; Digits : Integer) : string;
1582: function IntToHex(a: integer; b: integer): string;
1583: Function IntToHex64( Value : Int64; Digits : Integer) : string;
1584: function IntToHex64(Value: Int64; Digits: Integer): string)
1585: Function IntTo3Str( Value : Longint; separator: string) : string
1586: Function inttobool( aInt : LongInt) : Boolean
1587: function IntToStr(i: Int64): String;
1588: Function IntToStr64(Value: Int64): string)
1589: function IOResult: Integer
1590: Function IPv6AddressToStr(const AValue: TIdIPv6Address): string
1591: Function IsAccel(VK: Word; const Str: string): Boolean
1592: Function IsAddressInNetwork( Address : String) : Boolean
1593: Function IsAdministrator : Boolean
1594: Function IsAlias( const Name : string) : Boolean
1595: Function IsApplicationRunning( const AClassName, ApplName : string) : Boolean
1596: Function IsASCII( const AByte : Byte) : Boolean;
1597: Function IsASCIILDH( const AByte : Byte) : Boolean;
1598: Function IsAssembly(const FileName: string): Boolean;
1599: Function IsBcdNegative( const Bcd : TBcd) : Boolean
1600: Function IsBinary(const AChar : Char) : Boolean
1601: function IsConsole: Boolean)
1602: Function IsDelimiter( Delimiters, S : string; Index : Integer) : Boolean
1603: function IsDelimiter(const Delimiters: string; const S: string; Index: Integer): Boolean)
1604: Function IsDelphiDesignMode : boolean
1605: Function IsDelphiRunning : boolean
1606: Function IsDFAState : boolean
1607: Function IsDirectory( const FileName : string) : Boolean
1608: Function IsDomain( const S : String) : Boolean
1609: function IsDragObject(Sender: TObject): Boolean;
1610: Function IsEditing : Boolean
1611: Function ISEMPTY : BOOLEAN
1612: Function IsEqual( Value : TParameters) : Boolean
1613: Function ISEQUAL( VALUE : TPARAMS) : BOOLEAN
1614: function IsEqualGUID(const guid1, guid2: TGUID): Boolean)
1615: Function IsFirstNode : Boolean
1616: Function IsFloatZero( const X : Float) : Boolean
1617: Function IsFormatRegistered( Extension, AppID : string) : Boolean
1618: Function IsFormOpen(const FormName: string): Boolean;
1619: Function IsFQDN( const S : String) : Boolean
1620: Function IsGrayScale : Boolean
1621: Function IsHex( AChar : Char) : Boolean;
1622: Function IsHexString(const AString: string): Boolean;
1623: Function IsHostname( const S : String) : Boolean
1624: Function IsInfinite( const AValue : Double) : Boolean
1625: Function IsInLeapYear( const AValue : TDateTime) : Boolean
1626: Function IsInternet: boolean;
1627: Function IsLeadChar( ACh : Char) : Boolean
1628: Function IsLeapYear( Year : Word) : Boolean
1629: function IsLeapYear(Year: Word): Boolean)
1630: function IsLibrary: Boolean)
1631: Function ISLINE : BOOLEAN
1632: Function IsLinkedTo( DataSet : TDataSet) : Boolean
1633: Function ISLINKEDTO( DATASOURCE : TDATASOURCE) : BOOLEAN
1634: Function IsLiteralChar( const EditMask : string; Offset : Integer) : Boolean
1635: Function IsMatch( const Pattern, Text : string) : Boolean  //Grep like RegEx
1636: Function IsMainAppWindow( Wnd : HWND) : Boolean
1637: Function IsMediaPresentInDrive( Drive : Char) : Boolean
1638: function IsMemoryManagerSet: Boolean)
1639: Function IsMultiTableQuery( const SQL : WideString) : Boolean
1640: function IsMultiThread: Boolean)
1641: Function IsNumeric( AChar : Char) : Boolean;
1642: Function IsNumeric2( const AString : string) : Boolean;
1643: Function IsOctal( AChar : Char) : Boolean;
1644: Function IsOctalString(const AString: string) : Boolean;
1645: Function IsPathDelimiter( S : string; Index : Integer) : Boolean
1646: function IsPathDelimiter(const S: string; Index: Integer): Boolean)
1647: Function IsPM( const AValue : TDateTime) : Boolean
1648: Function IsPositiveFloatArray( const X : TDynFloatArray) : Boolean
1649: Function IsPrimeFactor( const F, N : Cardinal) : Boolean
1650: Function IsPrimeRM( N : Cardinal) : Boolean  //rabin miller
1651: Function IsPrimeTD( N : Cardinal) : Boolean  //trial division
1652: Function IsPrivilegeEnabled( const Privilege : string) : Boolean
1653: Function ISqrt( const I : Smallint) : Smallint
1654: Function IsReadOnly(const Filename: string): boolean;
1655: Function IsRectEmpty( const Rect : TRect) : Boolean
1656: function IsRectEmpty(const Rect: TRect): Boolean)
1657: Function IsRelativePrime( const X, Y : Cardinal) : Boolean
1658: Function ISRIGHTTOLEFT : BOOLEAN
1659: function IsRightToLeft: Boolean
1660: Function IsSameDay( const AValue, ABasis : TDateTime) : Boolean
1661: Function ISSEQUENCED : BOOLEAN
1662: Function IsSystemModule( const Module : HMODULE) : Boolean
```

```
1663: Function IsSystemResourcesMeterPresent : Boolean
1664: Function IsTCPPortOpen(dwPort : Word; ipAddressStr: String): boolean;
1665: Function IsToday( const AValue : TDateTime) : Boolean
1666: function IsToday(const AValue: TDateTime): Boolean;
1667: Function IsTopDomain( const AStr : string) : Boolean
1668: Function IsUTF8LeadByte( Lead : Char) : Boolean
1669: Function IsUTF8String( const s : UTF8String) : Boolean
1670: Function IsUTF8TrailByte( Lead : Char) : Boolean
1671: Function ISVALIDCHAR( INPUTCHAR : CHAR) : BOOLEAN
1672: Function IsValidDate( const AYear, AMonth, ADay : Word) : Boolean
1673: Function IsValidDateDay( const AYear, ADayOfYear : Word) : Boolean
1674: Function IsValidDateMonthWeek( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word) : Boolean
1675: Function IsValidDateTime(const AYear,AMonth, ADay,AHour,AMinute, ASecond, AMilliSecond: Word): Boolean
1676: Function IsValidDateWeek( const AYear, AWeekOfYear, ADayOfWeek : Word) : Boolean
1677: Function IsValidIdent( Ident : string) : Boolean
1678: function IsValidIdent(const Ident: string; AllowDots: Boolean): Boolean)
1679: Function IsValidIP( const S : String) : Boolean
1680: Function IsValidTime( const AHour, AMinute, ASecond, AMilliSecond : Word) : Boolean
1681: Function IsValidISBN( const ISBN : AnsiString) : Boolean
1682: Function IsVariantManagerSet: Boolean; //deprecated;
1683: Function IsVirtualPcGuest : Boolean;
1684: Function IsVmWareGuest : Boolean;
1685: Function IsVCLControl(Handle: HWnd): Boolean;
1686: Function IsWhiteString( const AStr : String) : Boolean
1687: Function IsWindowResponding( Wnd : HWND; Timeout : Integer) : Boolean
1688: Function IsWoW64: boolean;
1689: Function IsWin64: boolean;
1690: Function IsWow64String(var s: string): Boolean;
1691: Function IsWin64String(var s: string): Boolean;
1692: Function IsWindowsVista: boolean;
1693: Function isPowerof2(num: int64): boolean;
1694: Function powerOf2(exponent: integer): int64;
1695: function IsZero(const A: Extended; Epsilon: Extended): Boolean //overload;
1696: function IsZero1(const A: Double; Epsilon: Double): Boolean //overload;
1697: function IsZero2(const A: Single; Epsilon: Single): Boolean //overload;
1698: Function ItemAtPos( Pos : TPoint; IgnoreTabHeight : Boolean) : Integer
1699: function ITEMATPOS(POS:TPOINT;EXISTING:BOOLEAN):INTEGER
1700: Function ItemRect( Index : Integer) : TRect
1701: function ITEMRECT(INDEX:INTEGER):TRECT
1702: Function ItemWidth( Index : Integer) : Integer
1703: Function JavahashCode(val: string): Integer;
1704: Function JosephusG(n,k: integer; var graphout: string): integer;
1705: Function JulianDateToDateTime( const AValue : Double) : TDateTime
1706: Function JvMessageBox( const Text, Caption : string; Flags : DWORD) : Integer;
1707: Function JvMessageBox1( const Text : string; Flags : DWORD) : Integer;
1708: Function KeepAlive : Boolean
1709: Function KeysToShiftState(Keys: Word): TShiftState;
1710: Function KeyDataToShiftState(KeyData: Longint): TShiftState;
1711: Function KeyboardStateToShiftState2(const KeyboardState: TKeyboardState): TShiftState;
1712: Function KeyboardStateToShiftState: TShiftState; overload;
1713: Function Languages : TLanguages
1714: Function Last : TClass
1715: Function Last : TComponent
1716: Function Last : TObject
1717: Function LastDelimiter( Delimiters, S : string) : Integer
1718: function LastDelimiter(const Delimiters: string; const S: string): Integer)
1719: Function LastPos( const ASubstr : string; const AStr : string) : Integer
1720: Function Latitude2WGS84(lat: double): double;
1721: Function LCM(m,n:longint):longint;
1722: Function LCMJ( const X, Y : Cardinal) : Cardinal
1723: Function Ldexp( const X : Extended; const P : Integer) : Extended
1724: Function LeftStr( const AText : AnsiString; const ACount : Integer) : AnsiString;
1725: Function LeftStr( const AText : WideString; const ACount : Integer) : WideString;
1726: function Length: Integer;
1727: Procedure LetFileList(FileList: TStringlist; apath: string);
1728: function lengthmp3(mp3path: string):integer;
1729: Function LineInRect( const P1, P2 : TPoint; const Rect : TRect) : Boolean;
1730: Function LineInRect1( const P1, P2 : TFloatPoint; const Rect : TFloatRect) : Boolean;
1731: Function LineSegmentIntersection(const L1P1 : TFloatPoint; L1P2 : TFloatPoint; const L2P1 : TFloatPoint;
      L2P2 : TFloatPoint; var P : TFloatPoint) : Boolean
1732: function LineStart(Buffer, BufPos: PChar): PChar
1733: function LineStart(Buffer, BufPos: PChar): PChar)
1734: function ListSeparator: char;
1735: function Ln(x: Extended): Extended;
1736: Function LnXP1( const X : Extended) : Extended
1737: function Lo(vdat: word): byte;
1738: Function LoadCursor(hInstance: HINST; lpCursorName: PChar): HCURSOR
1739: Function LoadedModulesList( const List : TStrings; ProcessID : DWORD; HandlesOnly : Boolean) : Boolean
1740: Function LoadFileAsString( const FileName : string) : string
1741: Function LoadFromFile( const FileName : string) : TBitmapLoader
1742: Function LoadLibraryEx(LibName: PChar; hFile: Longint; Flags: Longint): Longint; stdcall;
1743: Function LoadPackage(const Name: string): HMODULE
1744: Function LoadResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : HGLOBAL
1745: Function LoadStr( Ident : Integer) : string
1746: Function LoadString(Instance: Longint; IDent: Integer; Buffer: PChar; Size: Integer): Integer; stdcall;
1747: Function LoadWideStr( Ident : Integer) : WideString
1748: Function LOCATE(const KEYFIELDS: String;const KEYVALUES:VARIANT;OPTIONS: TLOCATEOPTIONS) : BOOLEAN
1749: Function LockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
1750: Function LockServer( fLock : LongBool) : HResult
```

```
1751: Function LockVolume( const Volume : string; var Handle : THandle) : Boolean
1752: Function Log( const X : Extended) : Extended
1753: Function Log10( const X : Extended) : Extended
1754: Function Log2( const X : Extended) : Extended
1755: function LogBase10(X: Float): Float;
1756: Function LogBase2(X: Float): Float;
1757: Function LogBaseN(Base, X: Float): Float;
1758: Function LogN( const Base, X : Extended) : Extended
1759: Function LogOffOS : Boolean
1760: Function LoginDialog( const ADatabaseName : string; var AUserName, APassword : string) : Boolean
1761: Function LoginDialogEx(const ADatabaseName:string;var AUserName,
      APassword:string;NameReadOnly:Boolean):Boolean;
1762: Function LongDateFormat: string;
1763: function LongTimeFormat: string;
1764: Function LongWordToFourChar( ACardinal : LongWord) : string
1765: Function LOOKUP(const KEYFIELDS: String; const KEYVALUES: VARIANT; const RESULTFIELDS: String): VARIANT
1766: Function LookupName( const name : string) : TInAddr
1767: Function LookupService( const service : string) : Integer
1768: function Low: Int64;
1769: Function LowerCase( S : string) : string
1770: Function Lowercase(s : AnyString) : AnyString;
1771: Function LRot( const Value : Byte; const Count : TBitRange) : Byte;
1772: Function LRot1( const Value : Word; const Count : TBitRange) : Word;
1773: Function LRot2( const Value : Integer; const Count : TBitRange) : Integer;
1774: function MainInstance: longword
1775: function MainThreadID: longword
1776: Function Map(x, in_min, in_max, out_min, out_max: integer): integer;  //arduino
1777: Function mapMax(ax, in_min, in_max, out_min, out_max: integer): integer;
1778: Function MakeCanonicalIPv4Address( const AAddr : string) : string
1779: Function MakeCanonicalIPv6Address( const AAddr : string) : string
1780: Function MakeDIB( out Bitmap : PBitmapInfo) : Integer
1781: Function MakeDWordIntoIPv4Address( const ADWord : Cardinal) : string
1782: function MakeLong(A, B: Word): Longint)
1783: Function MakeTempFilename( const APath : String) : string
1784: Function MakeValidFileName( const Str : string) : string
1785: Function MakeValueMap( Enumeration : string; ToCds : Boolean) : string
1786: function MakeWord(A, B: Byte): Word)
1787: Function MakeYear4Digit( Year, Pivot : Integer) : Integer
1788: Function MapDateTime(const DateFormatType:string; DateFormat:string;Value:string;ToCds:Boolean): string
1789: Function MapValues( Mapping : string; Value : string) : string
1790: Function MaskDoFormatText( const EditMask : string; const Value : string; Blank : Char) : string
1791: Function MaskGetCharType( const EditMask : string; MaskOffset : Integer) : TMaskCharType
1792: Function MaskGetCurrentDirectives( const EditMask : string; MaskOffset : Integer) : TMaskDirectives
1793: Function MaskGetFldSeparator( const EditMask : string) : Integer
1794: Function MaskGetMaskBlank( const EditMask : string) : Char
1795: Function MaskGetMaskSave( const EditMask : string) : Boolean
1796: Function MaskIntlLiteralToChar( IChar : Char) : Char
1797: Function MaskOffsetToOffset( const EditMask : String; MaskOffset : Integer) : Integer
1798: Function MaskOffsetToWideOffset( const EditMask : String; MaskOffset : Integer) : Integer
1799: Function MaskString( Mask, Value : String) : String
1800: Function Match( const sString : string) : TniRegularExpressionMatchResul
1801: Function Match1( const sString : string; iStart : integer) : TniRegularExpressionMatchResult
1802: Function Matches( const Filename : string) : Boolean
1803: Function MatchesMask( const Filename, Mask : string) : Boolean
1804: Function MatchStr( const AText : string; const AValues : array of string) : Boolean
1805: Function MatchText( const AText : string; const AValues : array of string) : Boolean
1806: Function Max( AValueOne, AValueTwo : Integer) : Integer
1807: function Max(const x,y: Integer): Integer;
1808: Function Max1( const B1, B2 : Shortint) : Shortint;
1809: Function Max2( const B1, B2 : Smallint) : Smallint;
1810: Function Max3( const B1, B2 : Word) : Word;
1811: function Max3(const x,y,z: Integer): Integer;
1812: Function Max4( const B1, B2 : Integer) : Integer;
1813: Function Max5( const B1, B2 : Cardinal) : Cardinal;
1814: Function Max6( const B1, B2 : Int64) : Int64;
1815: Function Max64( const AValueOne, AValueTwo : Int64) : Int64
1816: Function MaxFloat( const X, Y : Float) : Float
1817: Function MaxFloatArray( const B : TDynFloatArray) : Float
1818: Function MaxFloatArrayIndex( const B : TDynFloatArray) : Integer
1819: function MaxIntValue(const Data: array of Integer):Integer)
1820: Function MaxJ( const B1, B2 : Byte) : Byte;
1821: function MaxPath: string;
1822: function MaxValue(const Data: array of Double): Double)
1823: Function MaxCalc( const Formula : string) : Extended  //math expression parser
1824: Procedure MaxCalcF( const Formula : string);   //out to console memo2
1825: function MD5(const fileName: string): string;
1826: Function Mean( const Data : array of Double) : Extended
1827: Function Median( const X : TDynFloatArray) : Float
1828: Function Memory : Pointer
1829: Function MemoryPos( const ASubStr : string; MemBuff : PChar; MemorySize : Integer) : Integer
1830: Function MessageBox(hndl: cardinal; text, caption: string; utype: cardinal): Integer;
1831: function MESSAGEBOX(TEXT,CAPTION:PCHAR;FLAGS:WORD):INTEGER
1832: Function MessageDlg(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
1833: Function MessageDlg1(const
      Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;DefaultButton:TMsgDlgBtn): Integer;
1834: Function MessageDlgPos(const Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint;X,
      Y:Integer):Integer;
1835: Function MessageDlgPos1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
      Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn) : Integer;
```

```
1836: Function MessageDlgPosHelp( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx :
      Longint; X, Y : Integer; const HelpFileName : string) : Integer;
1837: Function MessageDlgPosHelp1( const Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons; HelpCtx
      : Longint; X, Y : Integer; const HelpFileName : string; DefaultButton : TMsgDlgBtn) : Integer;
1838: Function MibToId( Mib : string) : string
1839: Function MidStr( const AText : AnsiString; const AStart, ACount : Integer) : AnsiString;
1840: Function MidStr( const AText : WideString; const AStart, ACount : Integer) : WideString;
1841: Function microsecondsToCentimeters(mseconds: longint): longint;  //340m/s speed of sound
1842: Function Micros(const Timer:THPTimer;const TimerRunning:Boolean):Int64//TypeS('THPTimer', 'Int64
1843: Function MIDIOut( DeviceID : Cardinal) : IJclMIDIOut
1844: Procedure GetMidiOutputs( const List : TStrings)
1845: Function MIDISingleNoteTuningData( Key : TMIDINote; Frequency : Single) : TSingleNoteTuningData
1846: Function MIDINoteToStr( Note : TMIDINote) : string
1847: Function WinMidiOut( DeviceID : Cardinal) : IJclWinMidiOut
1848: Procedure GetMidiOutputs( const List : TStrings)
1849: Procedure MidiOutCheck( Code : MMResult)
1850: Procedure MidiInCheck( Code : MMResult)
1851: Function MilliSecondOf( const AValue : TDateTime) : Word
1852: Function MilliSecondOfTheDay( const AValue : TDateTime) : LongWord
1853: Function MilliSecondOfTheHour( const AValue : TDateTime) : LongWord
1854: Function MilliSecondOfTheMinute( const AValue : TDateTime) : LongWord
1855: Function MilliSecondOfTheMonth( const AValue : TDateTime) : LongWord
1856: Function MilliSecondOfTheSecond( const AValue : TDateTime) : Word
1857: Function MilliSecondOfTheWeek( const AValue : TDateTime) : LongWord
1858: Function MilliSecondOfTheYear( const AValue : TDateTime) : Int64
1859: Function MilliSecondsBetween( const ANow, AThen : TDateTime) : Int64
1860: Function MilliSecondSpan( const ANow, AThen : TDateTime) : Double
1861: Function Micros( const Timer : THPTimer; const TimerRunning : Boolean) : Int64
1862: Function millis: int64;
1863: Function Min( AValueOne, AValueTwo : Integer) : Integer
1864: Function Min1( const B1, B2 : Shortint) : Shortint;
1865: Function Min2( const B1, B2 : Smallint) : Smallint;
1866: Function Min3( const B1, B2 : Word) : Word;
1867: Function Min4( const B1, B2 : Integer) : Integer;
1868: Function Min5( const B1, B2 : Cardinal) : Cardinal;
1869: Function Min6( const B1, B2 : Int64) : Int64;
1870: Function Min64( const AValueOne, AValueTwo : Int64) : Int64
1871: Function MinClientRect : TRect;
1872: Function MinClientRect1( IncludeScroller : Boolean) : TRect;
1873: Function MinClientRect2( TabCount : Integer; IncludeScroller : Boolean) : TRect;
1874: Function MinFloat( const X, Y : Float) : Float
1875: Function MinFloatArray( const B : TDynFloatArray) : Float
1876: Function MinFloatArrayIndex( const B : TDynFloatArray) : Integer
1877: Function MinimizeName( const Filename : string; Canvas : TCanvas; MaxLen : Integer) : string
1878: Function MinimizeName( const Filename : TFileName; Canvas : TCanvas; MaxLen : Integer) : TFileName
1879: function MinimizeName(const Filename: String; Canvas: TCanvas;MaxLen: Integer): TFileName
1880: Function MinIntValue( const Data : array of Integer) : Integer
1881: function MinIntValue(const Data: array of Integer):Integer)
1882: Function MinJ( const B1, B2 : Byte) : Byte;
1883: Function MinuteOf( const AValue : TDateTime) : Word
1884: Function MinuteOfTheDay( const AValue : TDateTime) : Word
1885: Function MinuteOfTheHour( const AValue : TDateTime) : Word
1886: Function MinuteOfTheMonth( const AValue : TDateTime) : Word
1887: Function MinuteOfTheWeek( const AValue : TDateTime) : Word
1888: Function MinuteOfTheYear( const AValue : TDateTime) : LongWord
1889: Function MinutesBetween( const ANow, AThen : TDateTime) : Int64
1890: Function MinuteSpan( const ANow, AThen : TDateTime) : Double
1891: Function MinValue( const Data : array of Double) : Double
1892: function MinValue(const Data: array of Double): Double)
1893: Function MixerLeftRightToArray( Left, Right : Cardinal) : TDynCardinalArray
1894: Function MMCheck( const MciError : MCIERROR; const Msg : string) : MCIERROR
1895: Function ModFloat( const X, Y : Float) : Float
1896: Function ModifiedJulianDateToDateTime( const AValue : Double) : TDateTime
1897: Function Modify( const Key : string; Value : Integer) : Boolean
1898: Function ModuleCacheID : Cardinal
1899: Function ModuleFromAddr( const Addr : Pointer) : HMODULE
1900: Function MonitorFromPoint( const Point : TPoint; MonitorDefault : TMonitorDefaultTo) : TMonitor
1901: Function MonitorFromRect( const Rect : TRect; MonitorDefault : TMonitorDefaultTo) : TMonitor
1902: Function MonitorFromWindow( const Handle : THandle; MonitorDefault : TMonitorDefaultTo) : TMonitor
1903: Function MonthOf( const AValue : TDateTime) : Word
1904: Function MonthOfTheYear( const AValue : TDateTime) : Word
1905: Function MonthsBetween( const ANow, AThen : TDateTime) : Integer
1906: Function MonthSpan( const ANow, AThen : TDateTime) : Double
1907: Function MonthStr( DateTime : TDateTime) : string
1908: Function MouseCoord( X, Y : Integer) : TGridCoord
1909: Function MOVEBY( DISTANCE : INTEGER) : INTEGER
1910: Function MoveFile( Source, Dest : string; Flags : FILEOP_FLAGS) : Boolean
1911: Function MoveNext : Boolean
1912: Function MSecsToTimeStamp( MSecs : Comp) : TTimeStamp
1913: function MSecsToTimeStamp(MSecs: Comp): TTimeStamp)
1914: Function Name : string
1915: Function NetPresentValue(const Rate:Extended;const CashFlows:array of
      Double;PaymentTime:TPaymentTime):Extended
1916: function NetworkVolume(DriveChar: Char): string
1917: Function NEWBOTTOMLINE : INTEGER
1918: Function NewCompareNode( Field : TField; Operator : TCANOperator; const Value : Variant) : PExprNode
1919: Function NEWITEM( const ACAPTION : String; ASHORTCUT : TSHORTCUT; ACHECKED, AENABLED : BOOLEAN; AONCLICK :
      TNOTIFYEVENT; HCTX : WORD; const ANAME : String) : TMENUITEM
1920: Function NEWLINE : TMENUITEM
```

```
1921: Function NEWMENU( OWNER : TCOMPONENT; const ANAME : STRING; ITEMS : array of TMenuItem) : TMAINMENU
1922: Function NewNode(Kind: TExprNodeKind; Operator:TCANOperator; const Data:Variant; Left,
      Right:PExprNode):PExprNode
1923: Function NEWPOPUPMENU(OWNER:TCOMPONENT;const ANAME:String; ALIGNMENT:TPOPUPALIGNMENT;AUTOPOPUP:BOOLEAN;
      const ITEMS : array of TCMENUITEM) : TPOPUPMENU
1924: Function NewState( eType : TniRegularExpressionStateType) : TniRegularExpressionState
1925: Function NEWSUBMENU(const ACAPT:String;HCTX:WORD;const ANAME:String;ITEMS:array of
      TMenuItem;AENABLED:BOOL): TMENUITEM
1926: Function NEWTOPLINE : INTEGER
1927: Function Next : TIdAuthWhatsNext
1928: Function NextCharIndex( S : String; Index : Integer) : Integer
1929: Function NextRecordSet : TCustomSQLDataSet
1930: Function NextRecordset( var RecordsAffected : Integer) : _Recordset
1931: Function NextSQLToken1( var p : WideChar; out Token : WideString; CurSection : TSQLToken) : TSQLToken;
1932: Function NextToken : Char
1933: Function nextToken : WideString
1934: function NextToken:Char
1935: Function Norm( const Data : array of Double) : Extended
1936: Function NormalizeAngle( const Angle : Extended) : Extended
1937: Function NormalizeBcd( const InBcd : TBcd; var OutBcd : TBcd; const Prec, Scale : Word) : Boolean
1938: Function NormalizeRect( const Rect : TRect) : TRect
1939: function NormalizeRect(const Rect: TRect): TRect;
1940: Function Now : TDateTime
1941: function Now2: tDateTime
1942: Function NumProcessThreads : integer
1943: Function NumThreadCount : integer
1944: Function NthDayOfWeek( const AValue : TDateTime) : Word
1945: Function NtProductType : TNtProductType
1946: Function NtProductTypeString : string
1947: function Null: Variant;
1948: Function NullPoint : TPoint
1949: Function NullRect : TRect
1950: Function Null2Blank(aString:String):String;
1951: Function NumberOfPeriods( const Rate : Extended; Payment : Extended; const PresentValue, FutureValue :
      Extended; PaymentTime : TPaymentTime) : Extended
1952: Function NumIP : integer
1953: function Odd(x: Longint): boolean;
1954: Function OffsetFromUTC : TDateTime
1955: Function OffsetPoint( const P, Offset : TPoint) : TPoint
1956: Function OffsetRect( var Rect : TRect; DX : Integer; DY : Integer) : Boolean
1957: function OffsetRect(var Rect: TRect; DX:Integer; DY:Integer): Boolean)
1958: Function OffsetToMaskOffset( const EditMask : string; Offset : Integer) : Integer
1959: Function OkToChangeFieldAlignment( AField : TField; Alignment : TAlignment) : Boolean
1960: Function OldBCDToCurr( const BCD : TBcd; var Curr : Currency) : Boolean
1961: Function OldCurrToBCD(const Curr:Currency; var BCD:TBcd; Precision:Integer;Decimals:Integer): Boolean
1962: function OpenBit:Integer
1963: Function OpenDatabase : TDatabase
1964: Function OpenDatabase( const DatabaseName : string) : TDatabase
1965: Function OpenGLColorToWinColor( const Red, Green, Blue : Float) : TColor
1966: Function OpenObject( Value : PChar) : Boolean;
1967: Function OpenObject1( Value : string) : Boolean;
1968: Function OpenSession( const SessionName : string) : TSession
1969: Function OpenVolume( const Drive : Char) : THandle
1970: function OrdFourByteToCardinal(AByte1, AByte2, AByte3, AByte4 : Byte): Cardinal
1971: Function OrdFourByteToLongWord( AByte1, AByte2, AByte3, AByte4 : Byte) : LongWord
1972: Function OrdToBinary( const Value : Byte) : string;
1973: Function OrdToBinary1( const Value : Shortint) : string;
1974: Function OrdToBinary2( const Value : Smallint) : string;
1975: Function OrdToBinary3( const Value : Word) : string;
1976: Function OrdToBinary4( const Value : Integer) : string;
1977: Function OrdToBinary5( const Value : Cardinal) : string;
1978: Function OrdToBinary6( const Value : Int64) : string;
1979: Function OSCheck( RetVal : Boolean) : Boolean
1980: Function OSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
1981: Function OSIdentToString( const OSIdent : DWORD) : string
1982: Function Output: Text)
1983: Function Overlay( ImageIndex : Integer; Overlay : TOverlay) : Boolean
1984: Function Owner : TCustomListView
1985: function Owner : TPersistent
1986: Function PadInputLiterals( const EditMask : String; const Value : string; Blank : Char) : string
1987: Function PadL( pStr : String; pLth : integer) : String
1988: Function Padl(s : AnyString;I : longInt) : AnyString;
1989: Function PadLCh( pStr : String; pLth : integer; pChr : char) : String
1990: Function PadR( pStr : String; pLth : integer) : String
1991: Function Padr(s : AnyString;I : longInt) : AnyString;
1992: Function PadRCh( pStr : String; pLth : integer; pChr : char) : String
1993: Function PadString( const AString : String; const ALen : Integer; const AChar : Char) : String
1994: Function Padz(s : AnyString;I : longInt) : AnyString;
1995: Function PaethPredictor( a, b, c : Byte) : Byte
1996: Function PARAMBYNAME( const VALUE : String) : TPARAM
1997: Function ParamByName( const Value : WideString) : TParameter
1998: Function ParamCount: Integer
1999: Function ParamsEncode( const ASrc : string) : string
2000: function ParamStr(Index: Integer): string)
2001: Function ParseDate( const DateStr : string) : TDateTime
2002: Function PARSESQL( SQL : String; DOCREATE : BOOLEAN) : String
2003: Function ParseSQL( SQL : WideString; DoCreate : Boolean) : WideString
2004: Function PathAddExtension( const Path, Extension : string) : string
2005: Function PathAddSeparator( const Path : string) : string
```

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

```
2090: Function Push( AItem : Pointer) : Pointer
2091: Function Push( AObject : TObject) : TObject
2092: Function Put1( AURL : string; const ASource : TStream) : string;
2093: Function Pythagoras( const X, Y : Extended) : Extended
2094: Function queryDLLInterface( var queryList : TStringList) : TStringList
2095: Function queryDLLInterfaceTwo( var queryList : TStringList) : TStringList
2096: Function QueryInterface(const IID: TGUID; out Obj): HResult, CdStdCall
2097: Function queryPerformanceCounter2(mse: int64): int64;
2098: //Function QueryPerformanceCounter(var lpPerformanceCount: Int64): LongBool; stdcall;
2099: //Function QueryPerformanceFrequency(mse: int64): boolean;
2100: Function QueryPerformanceCounter(var lcount: Int64): Boolean; stdcall;
2101: Function QueryPerformanceFrequency(var lfreq: int64): boolean; stdcall;
2102: Procedure QueryPerformanceCounter1(var aC: Int64);
2103: Function QueryPerformanceFrequency1(var freq: int64): boolean;
2104: Function Quote( const ACommand : String) : SmallInt
2105: Function QuotedStr( S : string) : string
2106: Function RadToCycle( const Radians : Extended) : Extended
2107: Function RadToDeg( const Radians : Extended) : Extended
2108: Function RadToDeg( const Value : Extended) : Extended;
2109: Function RadToDeg1( const Value : Double) : Double;
2110: Function RadToDeg2( const Value : Single) : Single;
2111: Function RadToGrad( const Radians : Extended) : Extended
2112: Function RadToGrad( const Value : Extended) : Extended;
2113: Function RadToGrad1( const Value : Double) : Double;
2114: Function RadToGrad2( const Value : Single) : Single;
2115: Function RandG( Mean, StdDev : Extended) : Extended
2116: function Random(const ARange: Integer): Integer;
2117: function random2(a: integer): double
2118: function RandomE: Extended;
2119: function RandomF: Extended;
2120: Function RandomFrom( const AValues : array of string) : string;
2121: Function RandomRange( const AFrom, ATo : Integer) : Integer
2122: function randSeed: longint
2123: Function RawToDataColumn( ACol : Integer) : Integer
2124: Function Read : Char
2125: Function Read( pv : Pointer; cb : Longint; pcbRead : PLongint) : HResult
2126: function Read(Buffer:String;Count:LongInt):LongInt
2127: Function ReadBinaryStream( const Section, Name : string; Value : TStream) : Integer
2128: Function ReadBool( const Section, Ident : string; Default : Boolean) : Boolean
2129: Function ReadCardinal( const AConvert : boolean) : Cardinal
2130: Function ReadChar : Char
2131: Function ReadClient( var Buffer, Count : Integer) : Integer
2132: Function ReadDate( const Section, Name : string; Default : TDateTime) : TDateTime
2133: Function ReadDateTime( const Section, Name : string; Default : TDateTime) : TDateTime
2134: Function ReadFloat( const Section, Name : string; Default : Double) : Double
2135: Function ReadFromStack(const ARaiseExceptfDisconnected:Bool;ATimeout:Int;const ARaiseExceptonTimeout:
      Boolean):Integer
2136: Function ReadInteger( const AConvert : boolean) : Integer
2137: Function ReadInteger( const Section, Ident : string; Default : Longint) : Longint
2138: Function ReadLn : string
2139: Function ReadLn( ATerminator : string; const ATimeout : Integer; AMaxLineLength : Integer) : string
2140: function Readln(question: string): string;
2141: Function ReadLnWait( AFailCount : Integer) : string
2142: Function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
2143: Function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
2144: Function ReadSmallInt( const AConvert : boolean) : SmallInt
2145: Function ReadString( const ABytes : Integer) : string
2146: Function ReadString( const Section, Ident, Default : string) : string
2147: Function ReadString( Count : Integer) : string
2148: Function ReadTime( const Section, Name : string; Default : TDateTime) : TDateTime
2149: Function ReadTimeStampCounter : Int64
2150: Function RebootOS : Boolean
2151: Function Receive( ATimeOut : Integer) : TReplyStatus
2152: Function ReceiveBuf( var Buf, Count : Integer) : Integer
2153: Function ReceiveLength : Integer
2154: Function ReceiveText : string
2155: Function ReceiveSerialData(var Data: TByteArray; DataSize: cardinal): cardinal
2156: Function ReceiveSerialText: string
2157: Function RecodeDate( const AValue : TDateTime; const AYear, AMonth, ADay : Word) : TDateTime
2158: Function RecodeDateTime(const AValue:TDateTime;const AYear,AMonth,ADay,AHr,AMin,ASec,
      AMilliSec:Word):TDateTime
2159: Function RecodeDay( const AValue : TDateTime; const ADay : Word) : TDateTime
2160: Function RecodeHour( const AValue : TDateTime; const AHour : Word) : TDateTime
2161: Function RecodeMilliSecond( const AValue : TDateTime; const AMilliSecond : Word) : TDateTime
2162: Function RecodeMinute( const AValue : TDateTime; const AMinute : Word) : TDateTime
2163: Function RecodeMonth( const AValue : TDateTime; const AMonth : Word) : TDateTime
2164: Function RecodeSecond( const AValue : TDateTime; const ASecond : Word) : TDateTime
2165: Function RecodeTime( const AValue: TDateTime;const AHour,AMinute,ASecond,AMilliSecond:Word):TDateTime
2166: Function RecodeYear( const AValue : TDateTime; const AYear : Word) : TDateTime
2167: Function Reconcile( const Results : OleVariant) : Boolean
2168: Function Rect( Left, Top, Right, Bottom : Integer) : TRect
2169: function Rect(ALeft: Integer; ATop: Integer; ARight: Integer; ABottom: Integer): TRect)
2170: Function Rect2( const ATopLeft, ABottomRight : TPoint) : TRect;
2171: Function RectAssign( const Left, Top, Right, Bottom : Integer) : TRect
2172: Function RectAssignPoints( const TopLeft, BottomRight : TPoint) : TRect
2173: Function RectBounds( const Left, Top, Width, Height : Integer) : TRect
2174: Function RectCenter( const R : TRect) : TPoint
2175: Function RectEqual( const R1, R2 : TRect) : Boolean
2176: Function RectHeight( const R : TRect) : Integer
```

```
2177: Function RectIncludesPoint( const R : TRect; const Pt : TPoint) : Boolean
2178: Function RectIncludesRect( const R1, R2 : TRect) : Boolean
2179: Function RectIntersection( const R1, R2 : TRect) : TRect
2180: Function RectIntersectRect( const R1, R2 : TRect) : Boolean
2181: Function RectIsEmpty( const R : TRect) : Boolean
2182: Function RectIsNull( const R : TRect) : Boolean
2183: Function RectIsSquare( const R : TRect) : Boolean
2184: Function RectIsValid( const R : TRect) : Boolean
2185: Function RectsAreValid( R : array of TRect) : Boolean
2186: Function RectUnion( const R1, R2 : TRect) : TRect
2187: Function RectWidth( const R : TRect) : Integer
2188: Function RedComponent( const Color32 : TColor32) : Integer
2189: Function Refresh : Boolean
2190: Function RefStringListCopy(aRefArray:TStringlist):TStringList;
2191: Function RegisterConversionFamily( const ADescription : string) : TConvFamily
2192: Function RegisterConversionType( AConvTypeInfo : TConvTypeInfo; out AType : TConvType) : Boolean;
2193: Function RegisterConversionType(const AFam:TConvFamil;const ADescr:string;const AFact:Double):TConvType
2194: Function RegistryRead(keyHandle: Longint; keyPath, myField: String): string;
2195: Function ReleaseDC(hdwnd: HWND; hdc: HDC): integer;
2196: Function ReleaseHandle : HBITMAP
2197: Function ReleaseHandle : HENHMETAFILE
2198: Function ReleaseHandle : HICON
2199: Function ReleasePalette : HPALETTE
2200: Function RemainderFloat( const X, Y : Float) : Float
2201: Function Remove( AClass : TClass) : Integer
2202: Function Remove( AComponent : TComponent) : Integer
2203: Function Remove( AItem : Integer) : Integer
2204: Function Remove( AItem : Pointer) : Pointer
2205: Function Remove( AItem : TObject) : TObject
2206: Function Remove( AObject : TObject) : Integer
2207: Function RemoveBackslash( const PathName : string) : string
2208: Function RemoveDF( aString : String) : String  //removes thousand separator
2209: Function RemoveDir( Dir : string) : Boolean
2210: function RemoveDir(const Dir: string): Boolean)
2211: Function RemoveDirectory(PathName: PChar): WordBool; stdcall;
2212: Function RemoveFileExt( const FileName : string) : string
2213: Function RemoveHeaderEntry( AHeader, AEntry : string) : string
2214: Function RenameFile( OldName, NewName : string) : Boolean
2215: function RenameFile(const OldName: string; const NewName: string): Boolean)
2216: Function ReplaceStr( const AText, AFromText, AToText : string) : string
2217: Function ReplaceText( const AText, AFromText, AToText : string) : string
2218: Function Replicate(c : char;I : longInt) : String;
2219: Function Request : TWebRequest
2220: Function ResemblesText( const AText, AOther : string) : Boolean
2221: Function Reset : Boolean
2222: function Reset2(mypath: string):string;
2223: Function ResInstLoad(Instance:THandle;ResType:TResType; const Name:string;MaskColor: TColor): Boolean
2224: Function ResourceLoad( ResType : TResType; const Name : string; MaskColor : TColor) : Boolean
2225: Function Response : TWebResponse
2226: Function ResumeSupported : Boolean
2227: Function RETHINKHOTKEYS : BOOLEAN
2228: Function RETHINKLINES : BOOLEAN
2229: Function Retrieve( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2230: Function RetrieveCurrentDir : string
2231: Function RetrieveDeltas( const cdsArray : array of TClientDataset) : Variant
2232: Function RetrieveHeader( const MsgNum : Integer; AMsg : TIdMessage) : Boolean
2233: Function RetrieveMailBoxSize : integer
2234: Function RetrieveMsgSize( const MsgNum : Integer) : Integer
2235: Function RetrieveProviders( const cdsArray : array of TClientDataset) : Variant
2236: Function RetrieveRaw( const MsgNum : Integer; const Dest : TStrings) : boolean
2237: Function ReturnMIMEType( var MediaType, EncType : String) : Boolean
2238: Function ReverseBits( Value : Byte) : Byte;
2239: Function ReverseBits1( Value : Shortint) : Shortint;
2240: Function ReverseBits2( Value : Smallint) : Smallint;
2241: Function ReverseBits3( Value : Word) : Word;
2242: Function ReverseBits4( Value : Cardinal) : Cardinal;
2243: Function ReverseBits4( Value : Integer) : Integer;
2244: Function ReverseBits5( Value : Int64) : Int64;
2245: Function ReverseBytes( Value : Word) : Word;
2246: Function ReverseBytes1( Value : Smallint) : Smallint;
2247: Function ReverseBytes2( Value : Integer) : Integer;
2248: Function ReverseBytes3( Value : Cardinal) : Cardinal;
2249: Function ReverseBytes4( Value : Int64) : Int64;
2250: Function ReverseString( const AText : string) : string
2251: Function ReverseDNSLookup(const IPAddrs:String;DNSServer:String;Timeout,Retries:Int;var
      HostName:String):Bool;
2252: Function Revert : HResult
2253: Function RGB(R,G,B: Byte): TColor;
2254: Function RGB2BGR( const Color : TColor) : TColor
2255: Function RGB2TColor( R, G, B : Byte) : TColor
2256: Function RGBToWebColorName( RGB : Integer) : string
2257: Function RGBToWebColorStr( RGB : Integer) : string
2258: Function RgbToHtml( Value : TColor) : string
2259: Function HtmlToRgb(const Value: string): TColor;
2260: Function RightStr( const AStr : String; Len : Integer) : String
2261: Function RightStr( const AText : AnsiString; const ACount : Integer) : AnsiString;
2262: Function RightStr( const AText : WideString; const ACount : Integer) : WideString;
2263: Function ROL( AVal : LongWord; AShift : Byte) : LongWord
2264: Function ROR( AVal : LongWord; AShift : Byte) : LongWord
```

```
2265: Function RotatePoint( Point : TFloatPoint; const Center : TFloatPoint; const Angle : Float) : TFloatPoint
2266: function RotatePoint(Point: TFloatPoint; const Center: TFloatPoint; const Angle: Double): TFloatPoint;
2267: Function Round(e : Extended) : Longint;
2268: Function Round64(e: extended): Int64;
2269: Function RoundAt( const Value : string; Position : SmallInt) : string
2270: Function RoundFrequency( const Frequency : Integer) : Integer
2271: Function RoundInt( Value : Integer; StepSize : Integer) : Integer
2272: Function RoundPoint( const X, Y : Double) : TPoint
2273: Function RoundRect( const ALeft, ATop, ARight, ABottom : Double) : TRect
2274: Function RowCount : Integer
2275: Function RowRequest( const Row : OleVariant; RequestType : Integer; var OwnerData : OleVariant): OleVariant
2276: Function RowRequest( Row : OleVariant; Options : TFetchOptions) : OleVariant
2277: Function RPos( const ASub, AIn : String; AStart : Integer) : Integer
2278: Function RRot( const Value : Byte; const Count : TBitRange) : Byte;
2279: Function RRot1( const Value : Word; const Count : TBitRange) : Word;
2280: Function RRot2( const Value : Integer; const Count : TBitRange) : Integer;
2281: Function RunDLL32(const ModuleNa,FuncName,CmdLine:string;WaitForCompletion:Bool;CmdShow:Integer):Boolean
2282: Function RunningProcessesList( const List : TStrings; FullPath : Boolean) : Boolean
2283: Function S_AddBackSlash( const ADirName : string) : string
2284: Function S_AllTrim( const cStr : string) : string
2285: Function S_AtRepl( const cAT, cStr, cRepl : string) : string
2286: Function S_Cut( const cStr : string; const iLen : integer) : string
2287: Function S_DecryptCRC32( const crc : string; StartKey, MultKey, AddKey : integer) : integer
2288: Function S_DirExists( const ADir : string) : Boolean
2289: Function S_Empty( const cStr : string) : boolean
2290: Function S_EncryptCRC32( const crc : LongWORD; StartKey, MultKey, AddKey : integer) : string
2291: Function S_LargeFontsActive : Boolean
2292: Function S_LimitDigits( AValue : Extended; ANumDigits : Integer) : Extended
2293: Function S_LTrim( const cStr : string) : string
2294: Function S_ReadNextTextLineFromStream( stream : TStream) : string
2295: Function S_RepeatChar( const iLen : integer; const AChar : Char) : String
2296: Function S_ReplFirst( const cAT, cStr, cRepl : string) : string
2297: Function S_RoundDecimal( AValue : Extended; APlaces : Integer) : Extended
2298: Function S_RTrim( const cStr : string) : string
2299: Function S_RTrimCopy( const cStr : string; iPos, iLen : integer) : string
2300: //Type TS_ShellExecuteCmd = (seCmdOpen,seCmdPrint,seCmdExplore);
2301: Function S_ShellExecute( aFilename : string; aParameters : string; aCommand : TS_ShellExecuteCmd) : string
2302: Function S_Space( const iLen : integer) : String
2303: Function S_StrBlanks( const cStr : string; const iLen : integer) : string
2304: Function S_StrBlanksCuttooLong( const cStr : string; const iLen : integer) : string
2305: Function S_StrCRC32( const Text : string) : LongWORD
2306: Function S_StrDecrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2307: Function S_StrEncrypt96( const InString : string; StartKey, MultKey, AddKey : Integer) : string
2308: Function S_StringtoUTF_8( const AString : string) : string
2309: Function S_StrLBlanks( const cStr : string; const iLen : integer) : string
2310: function S_StrToReal(const cStr: string; var R: Double): Boolean
2311: Function S_TokenEnd( cBuffer : PChar; lEmptyToken : boolean) : boolean
2312: Function S_TokenNext( cBuffer : PChar; lEmptyToken : boolean) : string
2313: Function S_UTF_8ToString( const AString : string) : string
2314: Function S_WBox( const AText : string) : integer
2315: Function SameDate( const A, B : TDateTime) : Boolean
2316: function SameDate(const A, B: TDateTime): Boolean;
2317: Function SameDateTime( const A, B : TDateTime) : Boolean
2318: function SameDateTime(const A, B: TDateTime): Boolean;
2319: Function SameFileName( S1, S2 : string) : Boolean
2320: Function SameText( S1, S2 : string) : Boolean
2321: function SameText(const S1: string; const S2: string): Boolean)
2322: Function SameTime( const A, B : TDateTime) : Boolean
2323: function SameTime(const A, B: TDateTime): Boolean;
2324: function SameValue(const A, B: Extended; Epsilon: Extended): Boolean //overload;
2325: function SameValue1(const A, B: Double; Epsilon: Double): Boolean //overload;
2326: function SameValue2(const A, B: Single; Epsilon: Single): Boolean //overload;
2327: Function SampleVariance( const X : TDynFloatArray) : Float
2328: Function Sar( const Value : Shortint; const Count : TBitRange) : Shortint;
2329: Function Sar1( const Value : Smallint; const Count : TBitRange) : Smallint;
2330: Function Sar2( const Value : Integer; const Count : TBitRange) : Integer;
2331: Function SaveToFile( const AFileName : TFileName) : Boolean
2332: Function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
2333: Function SaveAsExcel(aGrid: TStringGrid; aSheetName, aFileName: string; openexcel: boolean): Boolean;
2334: Function ScanF(const aformat: String; const args: array of const): string;
2335: Function SCREENTOCLIENT(POINT:TPOINT):TPOINT
2336: Function SearchBuf(Buf:PChar; BufLen:Integer;SelStart,SelLength:Integer;SearchString:String;Options:
      TStringSearchOptions):PChar
2337: Function SearchBuf2(Buf: String;SelStart,SelLength:Integer; SearchString:
      String;Options:TStringSearchOptions):Integer;
2338: function SearchRecattr: integer;
2339: function SearchRecExcludeAttr: integer;
2340: Function SearchRecFileSize64( const SearchRec : TSearchRec) : Int64
2341: function SearchRecname: string;
2342: function SearchRecsize: integer;
2343: function SearchRecTime: integer;
2344: Function Sec( const X : Extended) : Extended
2345: Function Secant( const X : Extended) : Extended
2346: Function SecH( const X : Extended) : Extended
2347: Function SecondOf( const AValue : TDateTime) : Word
2348: Function SecondOfTheDay( const AValue : TDateTime) : LongWord
2349: Function SecondOfTheHour( const AValue : TDateTime) : Word
2350: Function SecondOfTheMinute( const AValue : TDateTime) : Word
2351: Function SecondOfTheMonth( const AValue : TDateTime) : LongWord
```

```
2352: Function SecondOfTheWeek( const AValue : TDateTime) : LongWord
2353: Function SecondOfTheYear( const AValue : TDateTime) : LongWord
2354: Function SecondsBetween( const ANow, AThen : TDateTime) : Int64
2355: Function SecondSpan( const ANow, AThen : TDateTime) : Double
2356: Function SectionExists( const Section : string) : Boolean
2357: Function Seek( const KeyValues : Variant; SeekOption : TSeekOption) : Boolean
2358: Function Seek( dlibMove : Longint; dwOrigin : Longint; out libNewPosition : Largeint) : HResult
2359: function Seek(Offset:LongInt;Origin:Word):LongInt
2360: Function SelectDirectory( var Directory : string; Options : TSelectDirOpts; HelpCtx:Longint): Boolean;
2361: Function SelectDirectory1( const Caption : string; const Root : WideString; var Directory : string;
      Options : TSelectDirExtOpts; Parent : TWinControl) : Boolean;
2362: Function SelectImage( var AFileName : string; const Extensions, Filter : string) : Boolean
2363: function SendAppMessage(Msg: Cardinal; WParam, LParam: Longint): Longint
2364: Function SendBuf( var Buf, Count : Integer) : Integer
2365: Function SendCmd( const AOut : string; const AResponse : SmallInt) : SmallInt;
2366: Function SendCmd1( const AOut : string; const AResponse : array of SmallInt) : SmallInt;
2367: Function SendKey( AppName : string; Key : Char) : Boolean
2368: function SendMessage(hWnd: HWND; Msg: longword; wParam: longint; lParam: longint): Boolean;
2369: Function SendStream( AStream : TStream) : Boolean
2370: Function SendStreamThenDrop( AStream : TStream) : Boolean
2371: Function SendText( const S : string) : Integer
2372: Function SendSerialData(Data: TByteArray; DataSize: cardinal): cardinal
2373: Function SendSerialText(Data: String): cardinal
2374: Function Sent : Boolean
2375: Function ServicesFilePath: string
2376: Function SetAlpha( const Color32 : TColor32; NewAlpha : Integer) : TColor32
2377: Function SetBit( const Value : Byte; const Bit : TBitRange) : Byte;
2378: Function SetBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2379: Function SetBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2380: Function SetBit3( const Value : Word; const Bit : TBitRange) : Word;
2381: Function SetBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2382: Function SetBit4( const Value : Integer; const Bit : TBitRange) : Integer;
2383: Function SetBit5( const Value : Int64; const Bit : TBitRange) : Int64;
2384: Function SetClipboard( NewClipboard : TClipboard) : TClipboard
2385: Function SetColorBlue( const Color : TColor; const Blue : Byte) : TColor
2386: Function SetColorFlag( const Color : TColor; const Flag : Byte) : TColor
2387: Function SetColorGreen( const Color : TColor; const Green : Byte) : TColor
2388: Function SetColorRed( const Color : TColor; const Red : Byte) : TColor
2389: Function SetCurrentDir( Dir : string) : Boolean
2390: function SetCurrentDir(const Dir: string): Boolean)
2391: Function SetCurrentDirectory(PathName: PChar): WordBool; stdcall;
2392: Function SetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
2393: Function SetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
2394: Function SetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
2395: Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
2396: Function SetEndOfFile(Handle: Integer): LongBool; stdcall;
2397: Function SetEnvironmentVar( const Name, Value : string) : Boolean
2398: Function SetErrorProc( ErrorProc : TSocketErrorProc) : TSocketErrorProc
2399: Function SetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
2400: Function SetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
2401: Function SetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
2402: Function SetFileTimeStamp( const FileName : string; TimeStamp : Integer) : Boolean
2403: function SETFOCUSEDCONTROL(CONTROL:TWINCONTROL):BOOLEAN
2404: Function SetLocalTime( Value : TDateTime) : boolean
2405: Function SetPrecisionTolerance( NewTolerance : Float) : Float
2406: Function SetPrinter( NewPrinter : TPrinter) : TPrinter
2407: Function SetRGBValue( const Red, Green, Blue : Byte) : TColor
2408: Function SetSequence( S, Localizar, Substituir : shortstring) : shortstring
2409: Function SetSize( libNewSize : Longint) : HResult
2410: Function SetUserObjectFullAccess( hUserObject : THandle) : Boolean
2411: Function Sgn( const X : Extended) : Integer
2412: function SHA1(const fileName: string): string;
2413: function SHA256(astr: string; amode: char): string)
2414: function SHA512(astr: string; amode: char): string)
2415: Function ShareMemoryManager : Boolean
2416: function ShellExecute(hWnd:HWND;Operation,FileN,Parameters,Dir:string;ShowCmd:Integer):integer;stdcall;
2417: function Shellexecute2(hwnd: HWND; const FileName: string):integer; stdcall;
2418: Function ShellExecute3(aFilename: string; aParameters: string; aCommand:TS_ShellExecuteCmd): string;
2419: Function SHORTCUT( KEY : WORD; SHIFT : TSHIFTSTATE) : TSHORTCUT
2420: Function SHORTCUTTOTEXT( SHORTCUT : TSHORTCUT) : String
2421: function ShortDateFormat: string;
2422: Function ShortenString(const DC:HDC;const S:WideString;const Width:Int;const
      RTL:Bool;EllipsisWidth:Int):WideString
2423: function ShortTimeFormat: string;
2424: function SHOWMODAL:INTEGER
2425: function ShowWindow(C1: HWND; C2: integer): boolean;
2426: procedure ShowMemory     //in Dialog
2427: function ShowMemory2: string;
2428: Function ShutDownOS : Boolean
2429: Function Signe( const X, Y : Extended) : Extended
2430: Function Sign( const X : Extended) : Integer
2431: Function Sin(e : Extended) : Extended;
2432: Function sinc( const x : Double) : Double
2433: Function SinJ( X : Float) : Float
2434: Function Size( const AFileName : String) : Integer
2435: function SizeOf: Longint;
2436: Function SizeofResource( ModuleHandle : HMODULE; ResHandle : TResourceHandle) : Integer
2437: function SlashSep(const Path, S: String): String
2438: Function SLNDepreciation( const Cost, Salvage : Extended; Life : Integer) : Extended
```

```
2439: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
2440: Function SmallPoint(X, Y: Integer): TSmallPoint)
2441: Function Soundex( const AText : string; ALength : TSoundexLength) : string
2442: Function SoundexCompare( const AText, AOther : string; ALength : TSoundexLength) : Integer
2443: Function SoundexInt( const AText : string; ALength : TSoundexIntLength) : Integer
2444: Function SoundexProc( const AText, AOther : string) : Boolean
2445: Function SoundexSimilar( const AText, AOther : string; ALength : TSoundexLength) : Boolean
2446: Function SoundexWord( const AText : string) : Word
2447: Function SourcePos : Longint
2448: function SourcePos:LongInt
2449: Function Split0( Str : string; const substr : string) : TStringList
2450: Procedure SplitNameValue( const Line : string; var Name, Value : string)
2451: Function SQLRequiresParams( const SQL : WideString) : Boolean
2452: Function Sqr(e : Extended) : Extended;
2453: Function Sqrt(e : Extended) : Extended;
2454: Function StartIP : String
2455: Function StartPan( WndHandle : THandle; AControl : TControl) : Boolean
2456: Function StartOfADay( const AYear, AMonth, ADay : Word) : TDateTime;
2457: Function StartOfADay1( const AYear, ADayOfYear : Word) : TDateTime;
2458: Function StartOfAMonth( const AYear, AMonth : Word) : TDateTime
2459: Function StartOfAWeek( const AYear, AWeekOfYear : Word; const ADayOfWeek : Word) : TDateTime
2460: Function StartOfAYear( const AYear : Word) : TDateTime
2461: Function StartOfTheDay( const AValue : TDateTime) : TDateTime
2462: Function StartOfTheMonth( const AValue : TDateTime) : TDateTime
2463: Function StartOfTheWeek( const AValue : TDateTime) : TDateTime
2464: Function StartOfTheYear( const AValue : TDateTime) : TDateTime
2465: Function StartsStr( const ASubText, AText : string) : Boolean
2466: Function StartsText( const ASubText, AText : string) : Boolean
2467: Function StartsWith( const ANSIStr, APattern : String) : Boolean
2468: Function StartsWith( const str : string; const sub : string) : Boolean
2469: Function StartsWithACE( const ABytes : TIdBytes) : Boolean
2470: Function StatusString( StatusCode : Integer) : string
2471: Function StdDev( const Data : array of Double) : Extended
2472: Function Stop : Float
2473: Function StopCount( var Counter : TJclCounter) : Float
2474: Function StoreColumns : Boolean
2475: Function StrAfter( const sString : string; const sDelimiters : string) : string;
2476: Function StrAfter1( const sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2477: Function StrAlloc( Size : Cardinal) : PChar
2478: function StrAlloc(Size: Cardinal): PChar)
2479: Function StrBefore( const sString : string; const sDelimiters : string) : string;
2480: Function StrBefore1( const sString : string; const sDelimiters:string; out cDelimiter:char): string;
2481: Function StrBufSize( Str : PChar) : Cardinal
2482: function StrBufSize(const Str: PChar): Cardinal)
2483: Function StrByteType( Str : PChar; Index : Cardinal) : TMbcsByteType
2484: function StrByteType(Str: PChar; Index: Cardinal): TMbcsByteType)
2485: Function StrCat( Dest : PChar; Source : PChar) : PChar
2486: function StrCat(Dest: PChar; const Source: PChar): PChar)
2487: Function StrCharLength( Str : PChar) : Integer
2488: Function StrComp( Str1, Str2 : PChar) : Integer
2489: function StrComp(const Str1: PChar; const Str2: PChar): Integer)
2490: Function StrCopy( Dest : PChar; Source : PChar) : PChar
2491: function StrCopy(Dest: PChar; const Source: PChar): PChar)
2492: Function Stream_to_AnsiString( Source : TStream) : ansistring
2493: Function Stream_to_Base64( Source : TStream) : ansistring
2494: Function Stream_to_decimalbytes( Source : TStream) : string
2495: Function Stream2WideString( oStream : TStream) : WideString
2496: Function StreamtoAnsiString( Source : TStream) : ansistring
2497: Function StreamToByte( Source : TStream) : string
2498: Function StreamToDecimalbytes( Source : TStream) : string
2499: Function StreamtoOrd( Source : TStream) : string
2500: Function StreamToString( Source : TStream) : string
2501: Function StrECopy( Dest : PChar; Source : PChar) : PChar
2502: Function StrEmpty( const sString : string) : boolean
2503: Function StrEnd( Str : PChar) : PChar
2504: function StrEnd(const Str: PChar): PChar)
2505: Function StrFilter( const sString : string; xValidChars : TCharSet) : string
2506: Function StrFmt(Buffer, Format: PChar; const Args: array of const): PChar)
2507: Function StrGet(var S : String; I : Integer) : Char;
2508: Function StrGet2(S : String; I : Integer) : Char;
2509: Function StrHasPrefix( const sString : string; const sPrefix : string) : boolean
2510: Function StrHasSuffix( const sString : string; const sSuffix : string) : boolean
2511: Function StrHtmlDecode( const AStr : String) : String
2512: Function StrHtmlEncode( const AStr : String) : String
2513: Function StrToBytes(const Value: String): TBytes;
2514: Function StrIComp( Str1, Str2 : PChar) : Integer
2515: Function StringOfChar(c : char;I : longInt) : String;
2516: Function StringOfChar2( ch : WideChar; Count : Integer) : WideString;
2517: Function StringPad(InputStr,FillChar: String; StrLen:Integer; StrJustify:Boolean): String;
2518: Function StringRefCount(const s: String): integer;
2519: Function StringReplace( S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2520: Function JStringReplace( const S, OldPattern, NewPattern : string; Flags : TReplaceFlags) : string
2521: Function StringReplace(const SourceString, OldPattern,NewPattern:string; Flags:TReplaceFlags): string;
2522: Function StringRemove( const S, Pattern : string; Flags : TReplaceFlags) : string
2523: Function StringToBoolean( const Ps : string) : Boolean
2524: function StringToColor(const S: string): TColor)
2525: function StringToCursor(const S: string): TCursor;
2526: function StringToGUID(const S: string): TGUID)
2527: Function StringTokenizer( const str : string; const delim : string) : IStringTokenizer
```

```
2528: Function StringToStringArray( const str : string; const delim : string) : TStringDynArray
2529: Function StringWidth( S : string) : Integer
2530: Function StrInternetToDateTime( Value : string) : TDateTime
2531: Function StrIsDateTime( const Ps : string) : Boolean
2532: Function StrIsFloatMoney( const Ps : string) : Boolean
2533: Function StrIsInteger( const S : string) : Boolean
2534: Function StrLCat( Dest : PChar; Source : PChar; MaxLen : Cardinal) : PChar
2535: Function StrLComp( Str1, Str2 : PChar; MaxLen : Cardinal) : Integer
2536: Function StrLCopy( Dest : PChar; Source : PChar; MaxLen : Cardinal) : PChar
2537: Function StrLen( Str : PChar) : Cardinal
2538: function StrLen(const Str: PChar): Cardinal)
2539: Function StrLessPrefix( const sString : string; const sPrefix : string) : string
2540: Function StrLessSuffix( const sString : string; const sSuffix : string) : string
2541: Function StrLIComp( Str1, Str2 : PChar; MaxLen : Cardinal) : Integer
2542: Function StrLower( Str : PChar) : PChar
2543: Function StrMove( Dest : PChar; Source : PChar; Count : Cardinal) : PChar
2544: function StrMove(Dest: PChar; const Source: PChar; Count: Cardinal): PChar)
2545: Function StrNew( Str : PChar) : PChar
2546: function StrNew(const Str: PChar): PChar)
2547: Function StrNextChar( Str : PChar) : PChar
2548: Function StrPad( const sString : string; const sPad : string; const iLength : integer) : string
2549: Function StrParse( var sString : string; const sDelimiters : string) : string;
2550: Function StrParse1( var sString : string; const sDelimiters : string; out cDelimiter : char) : string;
2551: Function StrPas( Str : PChar) : string
2552: function StrPas(const Str: PChar): string)
2553: Function StrPCopy( Dest : PChar; Source : string) : PChar
2554: function StrPCopy(Dest: PChar; const Source: string): PChar)
2555: Function StrPLCopy( Dest : PChar; Source : string; MaxLen : Cardinal) : PChar
2556: Function StrPos( Str1, Str2 : PChar) : PChar
2557: Function StrScan(const Str: PChar; Chr: Char): PChar)
2558: Function StrRScan(const Str: PChar; Chr: Char): PChar)
2559: Function StrToBcd( const AValue : string) : TBcd
2560: Function StrToBool( S : string) : Boolean
2561: Function StrToBoolDef( S : string; Default : Boolean) : Boolean
2562: Function StrToCard( const AStr : String) : Cardinal
2563: Function StrToConv( AText : string; out AType : TConvType) : Double
2564: Function StrToCurr( S : string) : Currency;
2565: function StrToCurr(const S: string): Currency)
2566: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2567: Function StrToDate( S : string) : TDateTime;
2568: function StrToDate(const s: string): TDateTime;
2569: Function StrToDateDef( S : string; Default : TDateTime) : TDateTime;
2570: Function StrToDateTime( S : string) : TDateTime;
2571: function StrToDateTime(const S: string): TDateTime)
2572: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;
2573: Function StrToDay( const ADay : string) : Byte
2574: Function StrToFloat( S : string) : Extended;
2575: function StrToFloat(s: String): Extended;
2576: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2577: function StrToFloatDef(const S: string; const Default: Extended): Extended)
2578: Function StrToFloat( S : string) : Extended;
2579: Function StrToFloat2( S : string; FormatSettings : TFormatSettings) : Extended;
2580: Function StrToFloatDef( S : string; Default : Extended) : Extended;
2581: Function StrToFloatDef2(S: string; Default: Extended;FormatSettings:TFormatSettings): Extended;
2582: Function StrToCurr( S : string) : Currency;
2583: Function StrToCurr2( S : string; FormatSettings : TFormatSettings) : Currency;
2584: Function StrToCurrDef( S : string; Default : Currency) : Currency;
2585: Function StrToCurrDef2( S : string; Default : Currency; FormatSettings : TFormatSettings) : Currency;
2586: Function StrToTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
2587: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2588: Function StrToTimeDef2( S : string; Default : TDateTime; FormatSettings:TFormatSettings):TDateTime;
2589: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2590: Function StrToDateTime( S : string) : TDateTime;
2591: Function StrToDateTime2( S : string; FormatSettings : TFormatSettings) : TDateTime;
2592: Function StrToDateTimeDef( S : string; Default : TDateTime) : TDateTime;
2593: Function StrToFloatRegionalIndependent(aValue: String;aDecimalSymbol:Char;aDigitGroupSymbol:Char): Extended
2594: Function StrToInt( S : string) : Integer
2595: function StrToInt(s: String): Longint;
2596: Function StrToInt64( S : string) : Int64
2597: function StrToInt64(s: String): int64;
2598: Function StrToInt64Def( S : string; Default : Int64) : Int64
2599: function StrToInt64Def(const S: string; const Default: Int64):Int64)
2600: Function StrToIntDef( S : string; Default : Integer) : Integer
2601: function StrToIntDef(const S: string; Default: Integer): Integer)
2602: function StrToIntDef(s: String; def: Longint): Longint;
2603: Function StrToMonth( const AMonth : string) : Byte
2604: Function StrToTime( S : string) : TDateTime;
2605: function StrToTime(const S: string): TDateTime)
2606: Function StrToTimeDef( S : string; Default : TDateTime) : TDateTime;
2607: Function StrToWord( const Value : String) : Word
2608: Function StrToXmlDate( const DateStr : string; const Format : string) : string
2609: Function StrToXmlDateTime( const DateStr : string; const Format : string) : string
2610: Function StrToXmlTime( const TimeStr : string; const Format : string) : string
2611: Function StrUpper( Str : PChar) : PChar
2612: Function StuffString( const AText : string; AStart, ALength : Cardinal; const ASubText : string) : string
2613: Function Sum( const Data : array of Double) : Extended
2614: Function SumFloatArray( const B : TDynFloatArray) : Float
2615: Function SumInt( const Data : array of Integer) : Integer
2616: Function SumOfSquares( const Data : array of Double) : Extended
```

```
2617: Function SumPairProductFloatArray( const X, Y : TDynFloatArray) : Float
2618: Function SumSquareDiffFloatArray( const B : TDynFloatArray; Diff : Float) : Float
2619: Function SumSquareFloatArray( const B : TDynFloatArray) : Float
2620: Function Supports( CursorOptions : TCursorOptions) : Boolean
2621: Function SupportsClipboardFormat( AFormat : Word) : Boolean
2622: Function SwapWord(w : word): word)
2623: Function SwapInt(i : integer): integer)
2624: Function SwapLong(L : longint): longint)
2625: Function Swap(i : integer): integer)
2626: Function SYDDepreciation( const Cost, Salvage : Extended; Life, Period : Integer) : Extended
2627: Function SyncTime : Boolean
2628: Function SysErrorMessage( ErrorCode : Integer) : string
2629: function SysErrorMessage(ErrorCode: Integer): string)
2630: Function SystemTimeToDateTime( SystemTime : TSystemTime) : TDateTime
2631: function SystemTimeToDateTime(const SystemTime: TSystemTime): TDateTime;
2632: Function SysStringLen(const S: WideString): Integer; stdcall;
2633: Function TabRect( Index : Integer) : TRect
2634: Function Tan( const X : Extended) : Extended
2635: Function TaskMessageDlg(const Title,
      Msg:string;DlgType:TMsgDlgType;Buttons:TMsgDlgButtons;HelpCtx:Longint): Integer;
2636: Function TaskMessageDlg1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; DefaultButton : TMsgDlgBtn) : Integer;
2637: Function TaskMessageDlgPos( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; X, Y : Integer) : Integer;
2638: Function TaskMessageDlgPos1( const Title, Msg : string; DlgType : TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx : Longint; X, Y : Integer; DefaultButton : TMsgDlgBtn) : Integer;
2639: Function TaskMessageDlgPosHelp( const Title, Msg : string; DlgType : TMsgDlgType; Buttons :
      TMsgDlgButtons; HelpCtx : Longint; X, Y : Integer; const HelpFileName : string) : Integer;
2640: Function TaskMessageDlgPosHelp1(const Title, Msg:string;DlgType: TMsgDlgType; Buttons : TMsgDlgButtons;
      HelpCtx:Longint; X,Y: Integer;const HelpFileName:string;DefaultButton:TMsgDlgBtn): Integer;
2641: Function TenToY( const Y : Float) : Float
2642: Function TerminateApp( ProcessID : DWORD; Timeout : Integer) : TJclTerminateAppResult
2643: Function TerminateTask( Wnd : HWND; Timeout : Integer) : TJclTerminateAppResult
2644: Function TestBit( const Value : Byte; const Bit : TBitRange) : Boolean;
2645: Function TestBit2( const Value : Shortint; const Bit : TBitRange) : Boolean;
2646: Function TestBit3( const Value : Smallint; const Bit : TBitRange) : Boolean;
2647: Function TestBit4( const Value : Word; const Bit : TBitRange) : Boolean;
2648: Function TestBit5( const Value : Cardinal; const Bit : TBitRange) : Boolean;
2649: Function TestBit6( const Value : Integer; const Bit : TBitRange) : Boolean;
2650: Function TestBit7( const Value : Int64; const Bit : TBitRange) : Boolean;
2651: Function TestBits( const Value, Mask : Byte) : Boolean;
2652: Function TestBits1( const Value, Mask : Shortint) : Boolean;
2653: Function TestBits2( const Value, Mask : Smallint) : Boolean;
2654: Function TestBits3( const Value, Mask : Word) : Boolean;
2655: Function TestBits4( const Value, Mask : Cardinal) : Boolean;
2656: Function TestBits5( const Value, Mask : Integer) : Boolean;
2657: Function TestBits6( const Value, Mask : Int64) : Boolean;
2658: Function TestFDIVInstruction : Boolean
2659: function TestStreamFormat(Stream: TStream): TStreamOriginalFormat
2660: Function TextExtent( const Text : string) : TSize
2661: function TextHeight(Text: string): Integer;
2662: Function TextIsSame( const A1 : string; const A2 : string) : Boolean
2663: Function TextStartsWith( const S, SubS : string) : Boolean
2664: function TextToFloat(Buffer: PChar; var Value: Extended; ValueType: TFloatValue): Boolean)
2665: Function ConvInteger(i : integer):string;
2666: Function IntegerToText(i : integer):string;
2667: Function TEXTTOSHORTCUT( TEXT : String) : TSHORTCUT
2668: function TextWidth(Text: string): Integer;
2669: Function ThreadCount : integer
2670: function ThousandSeparator: char;
2671: Function Ticks : Cardinal
2672: Function Time : TDateTime
2673: function Time: TDateTime;
2674: function TimeGetTime: int64;
2675: Function TimeOf( const AValue : TDateTime) : TDateTime
2676: function TimeSeparator: char;
2677: function TimeStampToDateTime(const TimeStamp: TTimeStamp): TDateTime
2678: Function TimeStampToMSecs( TimeStamp : TTimeStamp) : Comp
2679: function TimeStampToMSecs(const TimeStamp: TTimeStamp): Comp)
2680: Function TimeToStr( DateTime : TDateTime) : string;
2681: function TimeToStr(const DateTime: TDateTime): string;
2682: Function TimeZoneBias : TDateTime
2683: Function ToCommon( const AValue : Double) : Double
2684: function ToCommon(const AValue: Double): Double;
2685: Function Today : TDateTime
2686: Function ToggleBit( const Value : Byte; const Bit : TBitRange) : Byte;
2687: Function ToggleBit1( const Value : Shortint; const Bit : TBitRange) : Shortint;
2688: Function ToggleBit2( const Value : Smallint; const Bit : TBitRange) : Smallint;
2689: Function ToggleBit3( const Value : Word; const Bit : TBitRange) : Word;
2690: Function ToggleBit4( const Value : Cardinal; const Bit : TBitRange) : Cardinal;
2691: Function ToggleBit5( const Value : Integer; const Bit : TBitRange) : Integer;
2692: Function ToggleBit6( const Value : Int64; const Bit : TBitRange) : Int64;
2693: function TokenComponentIdent:String
2694: Function TokenFloat : Extended
2695: function TokenFloat:Extended
2696: Function TokenInt : Longint
2697: function TokenInt:LongInt
2698: Function TokenString : string
2699: function TokenString:String
```

```
2700: Function TokenSymbolIs( const S : string) : Boolean
2701: function TokenSymbolIs(S:String):Boolean
2702: Function Tomorrow : TDateTime
2703: Function ToRightOf( const pc : TControl; piSpace : Integer) : Integer
2704: Function ToString : string
2705: Function TotalVariance( const Data : array of Double) : Extended
2706: Function Trace2( AURL : string) : string;
2707: Function TrackMenu( Button : TToolButton) : Boolean
2708: Function TRANSLATE( SRC, DEST : PCHAR; TOOEM : BOOLEAN) : INTEGER
2709: Function TranslateURI( const URI : string) : string
2710: Function TranslationMatchesLanguages( Exact : Boolean) : Boolean
2711: Function TransparentStretchBlt( DstDC :HDC;DstX,DstY,DstW,DstH:Integer;SrcDC:HDC;SrcX,SrcY,SrcW,
     SrcH:Integer;MaskDC : HDC; MaskX, MaskY : Integer) : Boolean
2712: Function Trim( S : string) : string;
2713: Function Trim( S : WideString) : WideString;
2714: Function Trim(s : AnyString) : AnyString;
2715: Function TrimAllOf( ATrim, AText : String) : String
2716: Function TrimLeft( S : string) : string;
2717: Function TrimLeft( S : WideString) : WideString;
2718: function TrimLeft(const S: string): string)
2719: Function TrimRight( S : string) : string;
2720: Function TrimRight( S : WideString) : WideString;
2721: function TrimRight(const S: string): string)
2722: function TrueBoolStrs: array of string
2723: Function Trunc(e : Extended) : Longint;
2724: Function Trunc64(e: extended): Int64;
2725: Function TruncPower( const Base, Exponent : Float) : Float
2726: Function TryConvTypeToFamily( const AFrom, ATo : TConvType; out AFamily : TConvFamily) : Boolean;
2727: Function TryConvTypeToFamily1( const AType : TConvType; out AFamily : TConvFamily) : Boolean;
2728: function TryEncodeDate(Year, Month, Day: Word; var Date: TDateTime): Boolean;
2729: Function TryEncodeDateDay( const AYear, ADayOfYear : Word; out AValue : TDateTime) : Boolean
2730: Function TryEncodeDateMonthWeek(const AY,AMonth,AWeekOfMonth,ADayOfWeek:Word;var AValue:TDateTime): Boolean
2731: Function TryEncodeDateTime(const AYear,AMonth,ADay,AHour,AMin,ASec,AMilliSecond:Word; out
     AValue:TDateTime):Boolean
2732: Function TryEncodeDateWeek(const AY,AWeekOfYear:Word;out AValue:TDateTime;const ADayOfWeek:Word): Boolean
2733: Function TryEncodeDayOfWeekInMonth(const AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word;out
     AVal:TDateTime):Bool
2734: function TryEncodeTime(Hour, Min, Sec, MSec: Word; var Time: TDateTime): Boolean;
2735: Function TryFloatToDateTime( Value : Extended; AResult : TDateTime) : Boolean
2736: Function TryJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2737: Function TryLock : Boolean
2738: Function TryModifiedJulianDateToDateTime( const AValue : Double; out ADateTime : TDateTime) : Boolean
2739: Function TryRecodeDateTime( const AValue : TDateTime; const AYear, AMonth, ADay, AHour, AMinute, ASecond,
     AMilliSecond : Word; out AResult : TDateTime) : Boolean
2740: Function TryStrToBcd( const AValue : string; var Bcd : TBcd) : Boolean
2741: Function TryStrToConv( AText : string; out AValue : Double; out AType : TConvType) : Boolean
2742: Function TryStrToDate( S : string; Value : TDateTime) : Boolean;
2743: Function TryStrToDateTime( S : string; Value : TDateTime) : Boolean;
2744: Function TryStrToTime( S : string; Value : TDateTime) : Boolean;
2745: Function TryStrToInt(const S: AnsiString; var I: Integer): Boolean;
2746: Function TryStrToInt64(const S: AnsiString; var I: Int64): Boolean;
2747: Function TwoByteToWord( AByte1, AByte2 : Byte) : Word
2748: Function TwoCharToWord( AChar1, AChar2 : Char) : Word
2749: Function TwoToY( const Y : Float) : Float
2750: Function UCS4StringToWideString( const S : UCS4String) : WideString
2751: Function UIDL( const ADest : TStrings; const AMsgNum : Integer) : Boolean
2752: function Unassigned: Variant;
2753: Function UndoLastChange( FollowChange : Boolean) : Boolean
2754: function UniCodeToStr(Value: string): string;
2755: Function UnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean
2756: function UnionRect(out Rect: TRect; const R1, R2: TRect): Boolean)
2757: Function UnixDateTimeToDelphiDateTime( UnixDateTime : Cardinal) : TDateTime
2758: Function UnixPathToDosPath( const Path : string) : string
2759: Function UnixToDateTime( const AValue : Int64) : TDateTime
2760: function UnixToDateTime(U: Int64): TDateTime;
2761: Function UnlockRegion( libOffset : Longint; cb : Largeint; dwLockType : Longint) : HResult
2762: Function UnlockResource( ResData : HGLOBAL) : LongBool
2763: Function UnlockVolume( var Handle : THandle) : Boolean
2764: Function UnMaskString( Mask, Value : String) : String
2765: function UpCase(ch : Char ) : Char;
2766: Function UpCaseFirst( const AStr : string) : string
2767: Function UpCaseFirstWord( const AStr : string) : string
2768: Function UpdateAction( Action : TBasicAction) : Boolean
2769: Function UpdateKind : TUpdateKind
2770: Function UPDATESTATUS : TUPDATESTATUS
2771: Function UpperCase( S : string) : string
2772: Function Uppercase(s : AnyString) : AnyString;
2773: Function URLDecode( ASrc : string) : string
2774: Function URLEncode( const ASrc : string) : string
2775: Function UseRightToLeftAlignment : Boolean
2776: Function UseRightToLeftAlignmentForField( const AField : TField; Alignment : TAlignment) : Boolean
2777: Function UseRightToLeftReading : Boolean
2778: Function UTF8CharLength( Lead : Char) : Integer
2779: Function UTF8CharSize( Lead : Char) : Integer
2780: Function UTF8Decode( const S : UTF8String) : WideString
2781: Function UTF8Encode( const WS : WideString) : UTF8String
2782: Function UTF8LowerCase( const S : UTF8string) : UTF8string
2783: Function Utf8ToAnsi( const S : UTF8String) : string
2784: Function Utf8ToAnsiEx( const S : UTF8String; const cp : integer) : string
```

```
2785: Function UTF8UpperCase( const S : UTF8string) : UTF8string
2786: Function ValidFieldIndex( FieldIndex : Integer) : Boolean
2787: Function ValidParentForm(control: TControl): TForm
2788: Function Value : Variant
2789: Function ValueExists( const Section, Ident : string) : Boolean
2790: Function ValueOf( const Key : string) : Integer
2791: Function ValueInSet(AValue: Variant; ASet: Variant): Boolean;
2792: Function VALUEOFKEY( const AKEY : VARIANT) : VARIANT
2793: Function VarArrayFromStrings( Strings : TStrings) : Variant
2794: Function VarArrayFromWideStrings( Strings : TWideStrings) : Variant
2795: Function VarArrayGet(var S : Variant; I : Integer) : Variant;
2796: Function VarFMTBcd : TVarType
2797: Function VarFMTBcdCreate1 : Variant;
2798: Function VarFMTBcdCreate2( const AValue : string; Precision, Scale : Word) : Variant;
2799: Function VarFMTBcdCreate3( const AValue : Double; Precision : Word; Scale : Word) : Variant;
2800: Function VarFMTBcdCreate4( const ABcd : TBcd) : Variant;
2801: Function Variance( const Data : array of Double) : Extended
2802: Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
2803: Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
2804: Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
2805: Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
2806: Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
2807: Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
2808: Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
2809: Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
2810: Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
2811: Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
2812: Function VariantNeg( const V1 : Variant) : Variant
2813: Function VariantNot( const V1 : Variant) : Variant
2814: Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
2815: Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
2816: Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
2817: Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
2818: Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
2819: function VarIsEmpty(const V: Variant): Boolean;
2820: Function VarIsFMTBcd( const AValue : Variant) : Boolean;
2821: function VarIsNull(const V: Variant): Boolean;
2822: Function VarToBcd( const AValue : Variant) : TBcd
2823: function VarType(const V: Variant): TVarType;
2824: Function VarType( const V : Variant) : TVarType
2825: Function VarAsType( const V : Variant; AVarType : TVarType) : Variant
2826: Function VarIsType( const V : Variant; AVarType : TVarType) : Boolean;
2827: Function VarIsType1( const V : Variant; const AVarTypes : array of TVarType) : Boolean;
2828: Function VarIsByRef( const V : Variant) : Boolean
2829: Function VarIsEmpty( const V : Variant) : Boolean
2830: Procedure VarCheckEmpty( const V : Variant)
2831: Function VarIsNull( const V : Variant) : Boolean
2832: Function VarIsClear( const V : Variant) : Boolean
2833: Function VarIsCustom( const V : Variant) : Boolean
2834: Function VarIsOrdinal( const V : Variant) : Boolean
2835: Function VarIsFloat( const V : Variant) : Boolean
2836: Function VarIsNumeric( const V : Variant) : Boolean
2837: Function VarIsStr( const V : Variant) : Boolean
2838: Function VarToStr( const V : Variant) : string
2839: Function VarToStrDef( const V : Variant; const ADefault : string) : string
2840: Function VarToWideStr( const V : Variant) : WideString
2841: Function VarToWideStrDef( const V : Variant; const ADefault : WideString) : WideString
2842: Function VarToDateTime( const V : Variant) : TDateTime
2843: Function VarFromDateTime( const DateTime : TDateTime) : Variant
2844: Function VarInRange( const AValue, AMin, AMax : Variant) : Boolean
2845: Function VarEnsureRange( const AValue, AMin, AMax : Variant) : Variant
2846: TVariantRelationship', '( vrEqual, vrLessThan, vrGreaterThan, vrNotEqual )
2847: Function VarSameValue( const A, B : Variant) : Boolean
2848: Function VarCompareValue( const A, B : Variant) : TVariantRelationship
2849: Function VarIsEmptyParam( const V : Variant) : Boolean
2850: Function VarIsError( const V : Variant; out AResult : HRESULT) : Boolean;
2851: Function VarIsError1( const V : Variant) : Boolean;
2852: Function VarAsError( AResult : HRESULT) : Variant
2853: Procedure VarCopyNoInd( var Dest : Variant; const Source : Variant)
2854: Function VarIsArray( const A : Variant) : Boolean;
2855: Function VarIsArray1( const A : Variant; AResolveByRef : Boolean) : Boolean;
2856: Function VarArrayCreate( const Bounds : array of Integer; AVarType : TVarType) : Variant
2857: Function VarArrayOf( const Values : array of Variant) : Variant
2858: Function VarArrayRef( const A : Variant) : Variant
2859: Function VarTypeIsValidArrayType( const AVarType : TVarType) : Boolean
2860: Function VarTypeIsValidElementType( const AVarType : TVarType) : Boolean
2861: Function VarArrayDimCount( const A : Variant) : Integer
2862: Function VarArrayLowBound( const A : Variant; Dim : Integer) : Integer
2863: Function VarArrayHighBound( const A : Variant; Dim : Integer) : Integer
2864: Function VarArrayLock( const A : Variant) : ___Pointer
2865: Procedure VarArrayUnlock( const A : Variant)
2866: Function VarArrayGet( const A : Variant; const Indices : array of Integer) : Variant
2867: Procedure VarArrayPut( var A : Variant; const Value : Variant; const Indices : array of Integer)
2868: Procedure DynArrayToVariant( var V : Variant; const DynArray : ___Pointer; TypeInfo : ___Pointer)
2869: Procedure DynArrayFromVariant( var DynArray : ___Pointer; const V : Variant; TypeInfo : ___Pointer)
2870: Function Unassigned : Variant
2871: Function Null : Variant
2872: Function VectorAdd( const V1, V2 : TFloatPoint) : TFloatPoint
2873: function VectorAdd(const V1,V2: TFloatPoint): TFloatPoint;
```

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

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

```
3050: Procedure CenterWindow( Wnd : HWND)
3051: Procedure ShadeRect( DC : HDC; const Rect : TRect)
3052: Procedure KillMessage( Wnd : HWND; Msg : Cardinal)
3053: Function DialogsToPixelsX( Dlgs : Word) : Word
3054: Function DialogsToPixelsY( Dlgs : Word) : Word
3055: Function PixelsToDialogsX( Pixs : Word) : Word
3056: Function PixelsToDialogsY( Pixs : Word) : Word
3057: {$IFDEF WIN32}
3058: procedure ShowMDIClientEdge(ClientHandle: THandle; ShowEdge: Boolean);
3059: function MakeVariant(const Values: array of Variant): Variant;
3060: {$ENDIF}
3061: function CreateRotatedFont(Font: TFont; Angle: Integer): HFONT;
3062: function MsgBox(const Caption, Text: string; Flags: Integer): Integer;
3063: function MsgDlg(const Msg:string; AType:TMsgDlgType; AButtons: TMsgDlgButtons; HelpCtx: Longint): Word;
3064: {$IFDEF CBUILDER}
3065: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): HWND;
3066: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
3067: {$ELSE}
3068: function FindPrevInstance(const MainFormClass, ATitle: string): HWND;
3069: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
3070: {$ENDIF CBUILDER}
3071: function IsForegroundTask: Boolean;
3072: procedure MergeForm(AControl: TWinControl; AForm: TForm; Align: TAlign; Show: Boolean);
3073: function GetAveCharSize(Canvas: TCanvas): TPoint;
3074: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
3075: procedure FreeUnusedOle;
3076: procedure Beep;
3077: function GetWindowsVersionJ: string;
3078: function LoadDLL(const LibName: string): THandle;
3079: function RegisterServer(const ModuleName: string): Boolean;
3080: {$IFNDEF WIN32}
3081: function IsLibrary: Boolean;
3082: {$ENDIF}
3083: { Gradient filling routine }
3084: type TFillDirection = (fdTopToBottom, fdBottomToTop, fdLeftToRight, fdRightToLeft);
3085: procedure GradientFillRect(Canvas: TCanvas; ARect: TRect; StartColor, EndColor: TColor; Direction:
      TFillDirection; Colors: Byte);
3086: { String routines }
3087: function GetEnvVar(const VarName: string): string;
3088: function AnsiUpperFirstChar(const S: string): string;
3089: function StringToPChar(var S: string): PChar;
3090: function StrPAlloc(const S: string): PChar;
3091: procedure SplitCommandLine(const CmdLine: string; var ExeName,Params: string);
3092: function DropT(const S: string): string;
3093: { Memory routines }
3094: function AllocMemo(Size: Longint): Pointer;
3095: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
3096: procedure FreeMemo(var fpBlock: Pointer);
3097: function GetMemoSize(fpBlock: Pointer): Longint;
3098: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
3099: {$IFNDEF COMPILER5_UP}
3100: procedure FreeAndNil(var Obj);
3101: {$ENDIF}
3102: // from PNGLoader
3103: Function OptimizeForPNG(Image:TLinearBitmap;QuantizationSteps:Integer;TransparentColor:TColor):Integer
3104: Procedure TransformRGB2LOCO( Image : TLinearBitmap)
3105: Procedure TransformLOCO2RGB( Image : TLinearBitmap)
3106: Procedure SortPalette( const Pal : TPalette; var ColorMap : TColorMap)
3107: Function DrawButtonFace( Canvas : TCanvas; const Client : TRect; BevelWidth : Integer; Style :
      TButtonStyle; IsRounded, IsDown, IsFocused : Boolean) : TRect  //TButtons
3108:   CL.AddDelphiFunction('Function IsAnAllResult( const AModalResult : TModalResult) : Boolean
3109:   Function InitWndProc( HWindow : HWnd; Message, WParam : Longint; LParam : Longint) : Longint
3110:  CL.AddConstantN('CTL3D_ALL','LongWord').SetUInt( $FFFF);
3111:  //Procedure ChangeBiDiModeAlignment( var Alignment : TAlignment)
3112:  //Function SendAppMessage( Msg : Cardinal; WParam, LParam : Longint) : Longint
3113:  //Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3114:  Procedure SetImeMode( hWnd : HWND; Mode : TImeMode)
3115:  Procedure SetImeName( Name : TImeName)
3116:  Function Win32NLSEnableIME( hWnd : HWND; Enable : Boolean) : Boolean
3117:  Function Imm32GetContext( hWnd : HWND) : HIMC
3118:  Function Imm32ReleaseContext( hWnd : HWND; hImc : HIMC) : Boolean
3119:  Function Imm32GetConversionStatus( hImc : HIMC; var Conversion, Sentence : longword) : Boolean
3120:  Function Imm32SetConversionStatus( hImc : HIMC; Conversion, Sentence : longword) : Boolean
3121:  Function Imm32SetOpenStatus( hImc : HIMC; fOpen : Boolean) : Boolean
3122: // Function Imm32SetCompositionWindow( hImc : HIMC; lpCompForm : PCOMPOSITIONFORM) : Boolean
3123:  //Function Imm32SetCompositionFont( hImc : HIMC; lpLogfont : PLOGFONTA) : Boolean
3124:  Function Imm32GetCompositionString(hImc:HIMC;dWord1:longword;lpBuf:string;dwBufLen:longint):Longint
3125:  Function Imm32IsIME( hKl : longword) : Boolean
3126:  Function Imm32NotifyIME( hImc : HIMC; dwAction, dwIndex, dwValue:longword):Boolean
3127:  Procedure DragDone( Drop : Boolean)
3128:
3129:
3130: //**************************************added  from jvjvclutils
3131: function CanvasMaxTextHeight(Canvas: TCanvas): Integer;
3132: function ReplaceComponentReference(This, NewReference: TComponent; var VarReference: TComponent): Boolean;
3133: procedure DrawLine(Canvas: TCanvas; X, Y, X2, Y2: Integer);
3134: function IsPositiveResult(Value: TModalResult): Boolean;
3135: function IsNegativeResult(Value: TModalResult): Boolean;
3136: function IsAbortResult(const Value: TModalResult): Boolean;
```

```
3137: function StripAllFromResult(const Value: TModalResult): TModalResult;
3138: // returns either BrightColor or DarkColor depending on the luminance of AColor
3139: // This function gives the same result (AFAIK) as the function used in Windows to
3140: // calculate the desktop icon text color based on the desktop background color
3141: function SelectColorByLuminance(AColor, DarkColor, BrightColor: TColor): TColor;
3142: type TJvHTMLCalcType = (htmlShow, htmlCalcWidth, htmlCalcHeight, htmlHyperLink);
3143:
3144: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3145:   const State: TOwnerDrawState; const Text: string; var Width: Integer;
3146:   CalcType: TJvHTMLCalcType;  MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3147:   var LinkName: string; Scale: Integer = 100); overload;
3148: procedure HTMLDrawTextEx(Canvas: TCanvas; Rect: TRect;
3149:   const State: TOwnerDrawState; const Text: string; var Width, Height: Integer;
3150:   CalcType: TJvHTMLCalcType;  MouseX, MouseY: Integer; var MouseOnLink: Boolean;
3151:   var LinkName: string; Scale: Integer = 100); overload;
3152: function HTMLDrawText(Canvas: TCanvas; Rect: TRect;
3153:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): string;
3154: function HTMLDrawTextHL(Canvas: TCanvas; Rect: TRect;
3155:   const State: TOwnerDrawState; const Text: string; MouseX, MouseY: Integer;
3156:   Scale: Integer = 100): string;
3157: function HTMLPlainText(const Text: string): string;
3158: function HTMLTextExtent(Canvas: TCanvas; Rect: TRect;
3159:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): TSize;
3160: function HTMLTextWidth(Canvas: TCanvas; Rect: TRect;
3161:   const State: TOwnerDrawState; const Text: string; Scale: Integer = 100): Integer;
3162: function HTMLTextHeight(Canvas: TCanvas; const Text: string; Scale: Integer = 100): Integer;
3163: function HTMLPrepareText(const Text: string): string;
3164:
3165: ****************************************** uPSI_JvAppUtils;
3166: Function GetDefaultSection( Component : TComponent) : string
3167: Procedure GetDefaultIniData(Control:TControl; var IniFileName, Section : string; UseRegistry : Boolean)
3168: Procedure GetDefaultIniData( Control : TControl; var IniFileName, Section : string)
3169: Function GetDefaultIniName : string
3170:  //'OnGetDefaultIniName','TOnGetDefaultIniName').SetString();
3171: Function GetDefaultIniRegKey : string
3172: Function FindForm( FormClass : TFormClass) : TForm
3173: Function FindShowForm( FormClass : TFormClass; const Caption : string) : TForm
3174: Function ShowDialog( FormClass : TFormClass) : Boolean
3175:  //Function InstantiateForm( FormClass : TFormClass; var Reference) : TForm
3176: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3177: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string; UseRegistry : Boolean)
3178: Procedure SaveMDIChildrenReg( MainForm : TForm; IniFile : TRegIniFile)
3179: Procedure SaveFormPlacement( Form : TForm; const IniFileName : string)
3180: Procedure RestoreFormPlacement( Form : TForm; const IniFileName : string)
3181: Function GetUniqueFileNameInDir( const Path, FileNameMask : string) : string
3182: Function StrToIniStr( const Str : string) : string
3183: Function IniStrToStr( const Str : string) : string
3184: Function IniReadString( IniFile : TObject; const Section, Ident, Default : string) : string
3185: Procedure IniWriteString( IniFile : TObject; const Section, Ident, Value : string)
3186: Function IniReadInteger( IniFile : TObject; const Section, Ident : string; Default : Longint) : Longint
3187: Procedure IniWriteInteger( IniFile : TObject; const Section, Ident : string; Value : Longint)
3188: Function IniReadBool( IniFile : TObject; const Section, Ident : string; Default : Boolean) : Boolean
3189: Procedure IniWriteBool( IniFile : TObject; const Section, Ident : string; Value : Boolean)
3190: Procedure IniReadSections( IniFile : TObject; Strings : TStrings)
3191: Procedure IniEraseSection( IniFile : TObject; const Section : string)
3192: Procedure IniDeleteKey( IniFile : TObject; const Section, Ident : string)
3193: Procedure AppBroadcast( Msg, wParam : Longint; lParam : Longint)
3194: Procedure AppBroadcast( Msg, wParam : Word; lParam : Longint)
3195: Procedure AppTaskbarIcons( AppOnly : Boolean)
3196: Procedure InternalSaveGridLayout( Grid : TCustomGrid; IniFile : TObject; const Section : string)
3197: Procedure InternalRestoreGridLayout( Grid : TCustomGrid; IniFile : TObject; const Section : string)
3198: Procedure InternalSaveMDIChildren( MainForm : TForm; IniFile : TObject)
3199: Procedure InternalRestoreMDIChildren( MainForm : TForm; IniFile : TObject)
3200: ****************************************** uPSI_JvDBUtils;
3201: Function CreateLocate( DataSet : TDataSet) : TJvLocateObject
3202: Function IsDataSetEmpty( DataSet : TDataSet) : Boolean
3203: Procedure RefreshQuery( Query : TDataSet)
3204: Function DataSetSortedSearch(DataSet:TDataSet;const Value,FieldName:string;CaseInsensitive:Bool):Boolean
3205: Function DataSetSectionName( DataSet : TDataSet) : string
3206: Procedure InternalSaveFields( DataSet : TDataSet; IniFile : TObject; const Section : string)
3207: Procedure InternalRestoreFields(DataSet:TDataSet;IniFile:TObject;const Section:string;RestoreVisible:Bool)
3208: Function DataSetLocateThrough(DataSet:TDataSet; const KeyFields: string; const KeyValues: Variant;
      Options: TLocateOptions) : Boolean
3209: Procedure SaveFields( DataSet : TDataSet; IniFile : TIniFile)
3210: Procedure RestoreFields( DataSet : TDataSet; IniFile : TIniFile; RestoreVisible : Boolean)
3211: Procedure AssignRecord( Source, Dest : TDataSet; ByName : Boolean)
3212: Function ConfirmDelete : Boolean
3213: Procedure ConfirmDataSetCancel( DataSet : TDataSet)
3214: Procedure CheckRequiredField( Field : TField)
3215: Procedure CheckRequiredFields( const Fields : array of TField)
3216: Function DateToSQL( Value : TDateTime) : string
3217: Function FormatSQLDateRange( Date1, Date2 : TDateTime; const FieldName : string) : string
3218: Function FormatSQLDateRangeEx( Date1, Date2 : TDateTime; const FieldName : string) : string
3219: Function FormatSQLNumericRange(const FieldName:string;LowVal,HighVal,LowEmpty,
      HighEmpty:Double;Inclusive:Bool):string
3220: Function StrMaskSQL( const Value : string) : string
3221: Function FormatSQLCondition(const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool):string
3222: Function FormatAnsiSQLCondition(const FieldName,Operator,Val:string;FieldType:TFieldType;Exact:Bool):string
3223: Procedure _DBError( const Msg : string)
```

```
3224:  Const('TrueExpr','String').SetString( '0=0
3225:  Const('sdfStandard16','String').SetString( ''''''mm''/''dd''/''yyyy''''''
3226:  Const('sdfStandard32','String').SetString( ''''''''dd/mm/yyyy''''''''
3227:  Const('sdfOracle','String').SetString( '"TO_DATE(''"dd/mm/yyyy"'', ''DD/MM/YYYY'')"
3228:  Const('sdfInterbase','String').SetString( '"CAST(''"mm"/"dd"/"yyyy"'' AS DATE)"
3229:  Const('sdfMSSQL','String').SetString( '"CONVERT(datetime, ''"mm"/"dd"/"yyyy"'', 103)"
3230:  AddTypeS('Largeint', 'Longint
3231:  addTypeS('TIFException', '(ErNoError, erCannotImport, erInvalidType, ErInternalError, '+
3232:    'erInvalidHeader, erInvalidOpcode, erInvalidOpcodeParameter, erNoMainProc, erOutOfGlobalVarsRange, '+
3233:    'erOutOfProcRange, ErOutOfRange, erOutOfStackRange, ErTypeMismatch, erUnexpectedEof, '+
3234:    'erVersionError, ErDivideByZero, ErMathError,erCouldNotCallProc, erOutofRecordRange, '+
3235:    'erOutOfMemory,erException,erNullPointerException,erNullVariantErrorerInterfaceNotSupportederError);
3236: (*----------------------------------------------------------------------------*)
3237: procedure SIRegister_JclIniFiles(CL: TPSPascalCompiler);
3238: begin
3239:  Function JIniReadBool( const FileName, Section, Line : string) : Boolean
3240:  Function JIniReadInteger( const FileName, Section, Line : string) : Integer
3241:  Function JIniReadString( const FileName, Section, Line : string) : string
3242:  Procedure JIniWriteBool( const FileName, Section, Line : string; Value : Boolean)
3243:  Procedure JIniWriteInteger( const FileName, Section, Line : string; Value : Integer)
3244:  Procedure JIniWriteString( const FileName, Section, Line, Value : string)
3245:  Procedure JIniReadStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3246:  Procedure JIniWriteStrings( IniFile : TCustomIniFile; const Section : string; Strings : TStrings)
3247: end;
3248:
3249: (* === compile-time registration functions === *)
3250: (*----------------------------------------------------------------------------*)
3251: procedure SIRegister_JclDateTime(CL: TPSPascalCompiler);
3252: begin
3253:  'UnixTimeStart','LongInt').SetInt( 25569);
3254:  Function JEncodeDate( const Year : Integer; Month, Day : Word) : TDateTime
3255:  Procedure JDecodeDate( Date : TDateTime; var Year, Month, Day : Word);
3256:  Procedure DecodeDate1( Date : TDateTime; var Year : Integer; var Month, Day : Word);
3257:  Procedure DecodeDate2( Date : TDateTime; var Year, Month, Day : Integer);
3258:  Function CenturyOfDate( const DateTime : TDateTime) : Integer
3259:  Function CenturyBaseYear( const DateTime : TDateTime) : Integer
3260:  Function DayOfDate( const DateTime : TDateTime) : Integer
3261:  Function MonthOfDate( const DateTime : TDateTime) : Integer
3262:  Function YearOfDate( const DateTime : TDateTime) : Integer
3263:  Function JDayOfTheYear( const DateTime : TDateTime; var Year : Integer) : Integer;
3264:  Function DayOfTheYear1( const DateTime : TDateTime) : Integer;
3265:  Function DayOfTheYearToDateTime( const Year, Day : Integer) : TDateTime
3266:  Function HourOfTime( const DateTime : TDateTime) : Integer
3267:  Function MinuteOfTime( const DateTime : TDateTime) : Integer
3268:  Function SecondOfTime( const DateTime : TDateTime) : Integer
3269:  Function GetISOYearNumberOfDays( const Year : Word) : Word
3270:  Function IsISOLongYear( const Year : Word) : Boolean;
3271:  Function IsISOLongYear1( const DateTime : TDateTime) : Boolean;
3272:  Function ISODayOfWeek( const DateTime : TDateTime) : Word
3273:  Function JISOWeekNumber( DateTime : TDateTime; var YearOfWeekNumber, WeekDay : Integer) : Integer;
3274:  Function ISOWeekNumber1( DateTime : TDateTime; var YearOfWeekNumber : Integer) : Integer;
3275:  Function ISOWeekNumber2( DateTime : TDateTime) : Integer;
3276:  Function ISOWeekToDateTime( const Year, Week, Day : Integer) : TDateTime
3277:  Function JIsLeapYear( const Year : Integer) : Boolean;
3278:  Function IsLeapYear1( const DateTime : TDateTime) : Boolean;
3279:  Function JDaysInMonth( const DateTime : TDateTime) : Integer
3280:  Function Make4DigitYear( Year, Pivot : Integer) : Integer
3281:  Function JMakeYear4Digit( Year, WindowsillYear : Integer) : Integer
3282:  Function JEasterSunday( const Year : Integer) : TDateTime // TDosDateTime', 'Integer
3283:  Function JFormatDateTime( Form : string; DateTime : TDateTime) : string
3284:  Function FATDatesEqual( const FileTime1, FileTime2 : Int64) : Boolean;
3285:  Function FATDatesEqual1( const FileTime1, FileTime2 : TFileTime) : Boolean;
3286:  Function HoursToMSecs( Hours : Integer) : Integer
3287:  Function MinutesToMSecs( Minutes : Integer) : Integer
3288:  Function SecondsToMSecs( Seconds : Integer) : Integer
3289:  Function TimeOfDateTimeToSeconds( DateTime : TDateTime) : Integer
3290:  Function TimeOfDateTimeToMSecs( DateTime : TDateTime) : Integer
3291:  Function DateTimeToLocalDateTime( DateTime : TDateTime) : TDateTime
3292:  Function LocalDateTimeToDateTime( DateTime : TDateTime) : TDateTime
3293:  Function DateTimeToDosDateTime( const DateTime : TDateTime) : TDosDateTime
3294:  Function JDateTimeToFileTime( DateTime : TDateTime) : TFileTime
3295:  Function JDateTimeToSystemTime( DateTime : TDateTime) : TSystemTime;
3296:  Procedure DateTimeToSystemTime1( DateTime : TDateTime; var SysTime : TSystemTime);
3297:  Function LocalDateTimeToFileTime( DateTime : TDateTime) : FileTime
3298:  Function DosDateTimeToDateTime( const DosTime : TDosDateTime) : TDateTime
3299:  Function JDosDateTimeToFileTime( DosTime : TDosDateTime) : TFileTime;
3300:  Procedure DosDateTimeToFileTime1( DTH, DTL : Word; FT : TFileTime);
3301:  Function DosDateTimeToSystemTime( const DosTime : TDosDateTime) : TSystemTime
3302:  Function DosDateTimeToStr( DateTime : Integer) : string
3303:  Function JFileTimeToDateTime( const FileTime : TFileTime) : TDateTime
3304:  Function FileTimeToLocalDateTime( const FileTime : TFileTime) : TDateTime
3305:  Function JFileTimeToDosDateTime( const FileTime : TFileTime) : TDosDateTime;
3306:  Procedure FileTimeToDosDateTime1( const FileTime : TFileTime; var Date, Time : Word);
3307:  Function JFileTimeToSystemTime( const FileTime : TFileTime) : TSystemTime;
3308:  Procedure FileTimeToSystemTime1( const FileTime : TFileTime; var ST : TSystemTime);
3309:  Function FileTimeToStr( const FileTime : TFileTime) : string
3310:  Function SystemTimeToDosDateTime( const SystemTime : TSystemTime) : TDosDateTime
3311:  Function JSystemTimeToFileTime( const SystemTime : TSystemTime) : TFileTime;
3312:  Procedure SystemTimeToFileTime1( const SystemTime : TSystemTime; FTime : TFileTime);
```

```
3313:  Function SystemTimeToStr( const SystemTime : TSystemTime) : string
3314:  Function CreationDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3315:  Function LastAccessDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3316:  Function LastWriteDateTimeOfFile( const Sr : TSearchRec) : TDateTime
3317:   TJclUnixTime32', 'Longword
3318:  Function JDateTimeToUnixTime( DateTime : TDateTime) : TJclUnixTime32
3319:  Function JUnixTimeToDateTime( const UnixTime : TJclUnixTime32) : TDateTime
3320:  Function FileTimeToUnixTime( const AValue : TFileTime) : TJclUnixTime32
3321:  Function UnixTimeToFileTime( const AValue : TJclUnixTime32) : TFileTime
3322:  Function JNullStamp : TTimeStamp
3323:  Function JCompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
3324:  Function JEqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
3325:  Function JIsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
3326:  Function TimeStampDOW( const Stamp : TTimeStamp) : Integer
3327:  Function FirstWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3328:  Function FirstWeekDay1( const Year, Month : Integer) : Integer;
3329:  Function LastWeekDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3330:  Function LastWeekDay1( const Year, Month : Integer) : Integer;
3331:  Function IndexedWeekDay( const Year, Month : Integer; Index : Integer) : Integer
3332:  Function FirstWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3333:  Function FirstWeekendDay1( const Year, Month : Integer) : Integer;
3334:  Function LastWeekendDay( const Year, Month : Integer; var DOW : Integer) : Integer;
3335:  Function LastWeekendDay1( const Year, Month : Integer) : Integer;
3336:  Function IndexedWeekendDay( const Year, Month : Integer; Index : Integer) : Integer
3337:  Function FirstDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3338:  Function LastDayOfWeek( const Year, Month, DayOfWeek : Integer) : Integer
3339:  Function IndexedDayOfWeek( const Year, Month, DayOfWeek, Index : Integer) : Integer
3340:   FindClass('TOBJECT'),'EJclDateTimeError
3341: end;
3342:
3343: procedure SIRegister_JclMiscel2(CL: TPSPascalCompiler);
3344: begin
3345:  Function SetDisplayResolution( const XRes, YRes : DWORD) : Longint
3346:  Function CreateDOSProcessRedirected( const CommandLine, InputFile, OutputFile : string) : Boolean
3347:  Function WinExec32( const Cmd : string; const CmdShow : Integer) : Boolean
3348:  Function WinExec32AndWait( const Cmd : string; const CmdShow : Integer) : Cardinal
3349:  Function WinExec32AndRedirectOutput(const Cmd: string; var Output: string; RawOutput:Boolean):Cardinal
3350:   TJclKillLevel', '( klNormal, klNoSignal, klTimeOut )
3351:  Function ExitWindows( ExitCode : Cardinal) : Boolean
3352:  Function LogOffOS( KillLevel : TJclKillLevel) : Boolean
3353:  Function PowerOffOS( KillLevel : TJclKillLevel) : Boolean
3354:  Function ShutDownOS( KillLevel : TJclKillLevel) : Boolean
3355:  Function RebootOS( KillLevel : TJclKillLevel) : Boolean
3356:  Function HibernateOS( Force, DisableWakeEvents : Boolean) : Boolean
3357:  Function SuspendOS( Force, DisableWakeEvents : Boolean) : Boolean
3358:  Function ShutDownDialog( const DialogMessage:string;TimeOut:DWORD;Force,Reboot:Boolean):Boolean;;
3359:  Function ShutDownDialog1(const MachineName,DialogMessage:string;TimeOut:DWORD;Force,Reboot:Bool):Bool;
3360:  Function AbortShutDown : Boolean;
3361:  Function AbortShutDown1( const MachineName : string) : Boolean;
3362:   TJclAllowedPowerOperation', '( apoHibernate, apoShutdown, apoSuspend )
3363:   TJclAllowedPowerOperations', 'set of TJclAllowedPowerOperation
3364:  Function GetAllowedPowerOperations : TJclAllowedPowerOperations
3365:   FindClass('TOBJECT'),'EJclCreateProcessError
3366:  Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
3367:  Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const
      Environment:PChar);
3368:   // with CL.Add(EJclCreateProcessError) do
3369:  end;
3370:
3371:
3372: procedure SIRegister_JclAnsiStrings(CL: TPSPascalCompiler);
3373: begin
3374:   //'AnsiSigns','Set').SetSet(['-', '+']);
3375:  'C1_UPPER','LongWord').SetUInt( $0001);
3376:  'C1_LOWER','LongWord').SetUInt( $0002);
3377:  'C1_DIGIT','LongWord').SetUInt( $0004);
3378:  'C1_SPACE','LongWord').SetUInt( $0008);
3379:  'C1_PUNCT','LongWord').SetUInt( $0010);
3380:  'C1_CNTRL','LongWord').SetUInt( $0020);
3381:  'C1_BLANK','LongWord').SetUInt( $0040);
3382:  'C1_XDIGIT','LongWord').SetUInt( $0080);
3383:  'C1_ALPHA','LongWord').SetUInt( $0100);
3384:   AnsiChar', 'Char
3385:  Function StrIsAlpha( const S : AnsiString) : Boolean
3386:  Function StrIsAlphaNum( const S : AnsiString) : Boolean
3387:  Function StrIsAlphaNumUnderscore( const S : AnsiString) : Boolean
3388:  Function StrContainsChars(const S:AnsiString;Chars:TSysCharSet; CheckAll : Boolean) : Boolean
3389:  Function StrConsistsOfNumberChars( const S : AnsiString) : Boolean
3390:  Function StrIsDigit( const S : AnsiString) : Boolean
3391:  Function StrIsSubset( const S : AnsiString; const ValidChars : TSysCharSet) : Boolean
3392:  Function StrSame( const S1, S2 : AnsiString) : Boolean
3393:  //Function StrCenter( const S : AnsiString; L : Integer; C : AnsiChar) : AnsiString
3394:  Function StrCharPosLower( const S : AnsiString; CharPos : Integer) : AnsiString
3395:  Function StrCharPosUpper( const S : AnsiString; CharPos : Integer) : AnsiString
3396:  Function StrDoubleQuote( const S : AnsiString) : AnsiString
3397:  Function StrEnsureNoPrefix( const Prefix, Text : AnsiString) : AnsiString
3398:  Function StrEnsureNoSuffix( const Suffix, Text : AnsiString) : AnsiString
3399:  Function StrEnsurePrefix( const Prefix, Text : AnsiString) : AnsiString
3400:  Function StrEnsureSuffix( const Suffix, Text : AnsiString) : AnsiString
```

```
3401:  Function StrEscapedToString( const S : AnsiString) : AnsiString
3402:  Function JStrLower( const S : AnsiString) : AnsiString
3403:  Procedure StrLowerInPlace( var S : AnsiString)
3404:  ///Procedure StrLowerBuff( S : PAnsiChar)
3405:  Procedure JStrMove( var Dest:AnsiString; const Source:AnsiString;const ToIndex,FromIndex,Count:Integer;
3406:  Function StrPadLeft( const S : AnsiString; Len : Integer; C : AnsiChar) : AnsiString
3407:  Function StrPadRight( const S : AnsiString; Len : Integer; C : AnsiChar) : AnsiString
3408:  Function StrProper( const S : AnsiString) : AnsiString
3409:  //Procedure StrProperBuff( S : PAnsiChar)
3410:  Function StrQuote( const S : AnsiString; C : AnsiChar) : AnsiString
3411:  Function StrRemoveChars( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3412:  Function StrKeepChars( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3413:  Procedure JStrReplace(var S:AnsiString; const Search, Replace : AnsiString; Flags : TReplaceFlags)
3414:  Function StrReplaceChar( const S : AnsiString; const Source, Replace : AnsiChar) : AnsiString
3415:  Function StrReplaceChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString
3416:  Function StrReplaceButChars(const S:AnsiString;const Chars:TSysCharSet;Replace:AnsiChar):AnsiString;
3417:  Function StrRepeat( const S : AnsiString; Count : Integer) : AnsiString
3418:  Function StrRepeatLength( const S : AnsiString; const L : Integer) : AnsiString
3419:  Function StrReverse( const S : AnsiString) : AnsiString
3420:  Procedure StrReverseInPlace( var S : AnsiString)
3421:  Function StrSingleQuote( const S : AnsiString) : AnsiString
3422:  Function StrSmartCase( const S : AnsiString; Delimiters : TSysCharSet) : AnsiString
3423:  Function StrStringToEscaped( const S : AnsiString) : AnsiString
3424:  Function StrStripNonNumberChars( const S : AnsiString) : AnsiString
3425:  Function StrToHex( const Source : AnsiString) : AnsiString
3426:  Function StrTrimCharLeft( const S : AnsiString; C : AnsiChar) : AnsiString
3427:  Function StrTrimCharsLeft( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3428:  Function StrTrimCharRight( const S : AnsiString; C : AnsiChar) : AnsiString
3429:  Function StrTrimCharsRight( const S : AnsiString; const Chars : TSysCharSet) : AnsiString
3430:  Function StrTrimQuotes( const S : AnsiString) : AnsiString
3431:  Function JStrUpper( const S : AnsiString) : AnsiString
3432:  Procedure StrUpperInPlace( var S : AnsiString)
3433:  //Procedure StrUpperBuff( S : PAnsiChar)
3434:  Function StrOemToAnsi( const S : AnsiString) : AnsiString
3435:  Function StrAnsiToOem( const S : AnsiString) : AnsiString
3436:  Procedure StrAddRef( var S : AnsiString)
3437:  Function StrAllocSize( const S : AnsiString) : Longint
3438:  Procedure StrDecRef( var S : AnsiString)
3439:  //Function StrLen( S : PAnsiChar) : Integer
3440:  Function StrLength( const S : AnsiString) : Longint
3441:  Function StrRefCount( const S : AnsiString) : Longint
3442:  Procedure StrResetLength( var S : AnsiString)
3443:  Function StrCharCount( const S : AnsiString; C : AnsiChar) : Integer
3444:  Function StrCharsCount( const S : AnsiString; Chars : TSysCharSet) : Integer
3445:  Function StrStrCount( const S, SubS : AnsiString) : Integer
3446:  Function StrCompare( const S1, S2 : AnsiString) : Integer
3447:  Function StrCompareRange( const S1, S2 : AnsiString; const Index, Count : Integer) : Integer
3448:  //Function StrFillChar( const C : AnsiChar; Count : Integer) : AnsiString;
3449:  Function StrFillChar1( const C : Char; Count : Integer) : AnsiString;
3450:  Function StrFind( const Substr, S : AnsiString; const Index : Integer) : Integer
3451:  //Function StrHasPrefix( const S : AnsiString; const Prefixes : array of AnsiString) : Boolean
3452:  Function StrIndex( const S : AnsiString; const List : array of AnsiString) : Integer
3453:  Function StrILastPos( const SubStr, S : AnsiString) : Integer
3454:  Function StrIPos( const SubStr, S : AnsiString) : Integer
3455:  Function StrIsOneOf( const S : AnsiString; const List : array of AnsiString) : Boolean
3456:  Function StrLastPos( const SubStr, S : AnsiString) : Integer
3457:  Function StrMatch( const Substr, S : AnsiString; const Index : Integer) : Integer
3458:  Function StrMatches( const Substr, S : AnsiString; const Index : Integer) : Boolean
3459:  Function StrNIPos( const S, SubStr : AnsiString; N : Integer) : Integer
3460:  Function StrNPos( const S, SubStr : AnsiString; N : Integer) : Integer
3461:  Function StrPrefixIndex( const S : AnsiString; const Prefixes : array of AnsiString) : Integer
3462:  Function StrSearch( const Substr, S : AnsiString; const Index : Integer) : Integer
3463:  //Function StrAfter( const SubStr, S : AnsiString) : AnsiString
3464:  //Function StrBefore( const SubStr, S : AnsiString) : AnsiString
3465:  Function StrBetween( const S : AnsiString; const Start, Stop : AnsiChar) : AnsiString
3466:  Function StrChopRight( const S : AnsiString; N : Integer) : AnsiString
3467:  Function StrLeft( const S : AnsiString; Count : Integer) : AnsiString
3468:  Function StrMid( const S : AnsiString; Start, Count : Integer) : AnsiString
3469:  Function StrRestOf( const S : AnsiString; N : Integer) : AnsiString
3470:  Function StrRight( const S : AnsiString; Count : Integer) : AnsiString
3471:  Function CharEqualNoCase( const C1, C2 : AnsiChar) : Boolean
3472:  Function CharIsAlpha( const C : AnsiChar) : Boolean
3473:  Function CharIsAlphaNum( const C : AnsiChar) : Boolean
3474:  Function CharIsBlank( const C : AnsiChar) : Boolean
3475:  Function CharIsControl( const C : AnsiChar) : Boolean
3476:  Function CharIsDelete( const C : AnsiChar) : Boolean
3477:  Function CharIsDigit( const C : AnsiChar) : Boolean
3478:  Function CharIsLower( const C : AnsiChar) : Boolean
3479:  Function CharIsNumberChar( const C : AnsiChar) : Boolean
3480:  Function CharIsPrintable( const C : AnsiChar) : Boolean
3481:  Function CharIsPunctuation( const C : AnsiChar) : Boolean
3482:  Function CharIsReturn( const C : AnsiChar) : Boolean
3483:  Function CharIsSpace( const C : AnsiChar) : Boolean
3484:  Function CharIsUpper( const C : AnsiChar) : Boolean
3485:  Function CharIsWhiteSpace( const C : AnsiChar) : Boolean
3486:  Function CharType( const C : AnsiChar) : Word
3487:  Function CharHex( const C : AnsiChar) : Byte
3488:  Function CharLower( const C : AnsiChar) : AnsiChar
3489:  Function CharUpper( const C : AnsiChar) : AnsiChar
```

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

```
3576: Procedure ExportDataSet( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
      AsciiCharSet : string; AsciiDelimited : Boolean; MaxRecordCount : Longint)
3577: Procedure ExportDataSetEx( Source : TBDEDataSet; DestTable : TTable; TableType : TTableType; const
      AsciiCharSet:string;AsciiDelimited:Boolean;AsciiDelimiter,AsciiSeparator:Char;MaxRecordCount:Longint);
3578: Procedure
      ImportDataSet(Source:TBDEDataSet;DestTable:TTable;MaxRecordCount:Longint;Mappings:TStrings;Mode:TBatchMode);
3579: Procedure InitRSRUN(Database: TDatabase;const ConName:string; ConType:Integer;const ConServer:string);
3580: ******************************************uPSI_JvDateUtil;
3581: function CurrentYear: Word;
3582: function IsLeapYear(AYear: Integer): Boolean;
3583: function DaysPerMonth(AYear, AMonth: Integer): Integer;
3584: function FirstDayOfPrevMonth: TDateTime;
3585: function LastDayOfPrevMonth: TDateTime;
3586: function FirstDayOfNextMonth: TDateTime;
3587: function ExtractDay(ADate: TDateTime): Word;
3588: function ExtractMonth(ADate: TDateTime): Word;
3589: function ExtractYear(ADate: TDateTime): Word;
3590: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
3591: function IncDay(ADate: TDateTime; Delta: Integer): TDateTime;
3592: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
3593: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
3594: function ValidDate(ADate: TDateTime): Boolean;
3595: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
3596: function MonthsBetween(Date1, Date2: TDateTime): Double;
3597: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
3598: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
3599: function DaysBetween(Date1, Date2: TDateTime): Longint;
3600: { The same as previous but if Date2 < Date1 result = 0 }
3601: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
3602: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
3603: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
3604: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
3605: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
3606: function CutTime(ADate: TDateTime): TDateTime; { Set time to 00:00:00:00 }
3607: { String to date conversions }
3608: function GetDateOrder(const DateFormat: string): TDateOrder;
3609: function MonthFromName(const S: string; MaxLen: Byte): Byte;
3610: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
3611: function StrToDateFmt(const DateFormat, S: string): TDateTime;
3612: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
3613: function DefDateFormat(FourDigitYear: Boolean): string;
3614: function DefDateMask(BlanksChar: Char; FourDigitYear: Boolean): string;
3615: ----------------------------------------------------------------------------
3616: ******************************** JvUtils;*****************************
3617: { GetWordOnPos returns Word from string, S, on the cursor position, P}
3618: function GetWordOnPos(const S: string; const P: Integer): string;
3619: { GetWordOnPosEx work like GetWordOnPos function, also returns Word position in iBeg, iEnd variables }
3620: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3621: { SubStr returns substring from string, S, separated with Separator string}
3622: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3623: { SubStrEnd same to previous function but Index numerated from the end of string }
3624: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3625: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
3626: function SubWord(P: PChar; var P2: PChar): string;
3627: { NumberByWord returns the text representation of
3628:   the number, N, in normal russian language. Was typed from Monitor magazine }
3629: function NumberByWord(const N: Longint): string;
3630: //  function CurrencyByWord(Value : Currency) : string; GetLineByPos returns Line number, there
3631:   //the symbol Pos is pointed. Lines separated with #13 symbol }
3632: function GetLineByPos(const S: string; const Pos: Integer): Integer;
3633: { GetXYByPos is same to previous function, but returns X position in line too}
3634: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3635: { ReplaceString searches for all substrings, OldPattern,in a string, S, and replaces them with NewPattern }
3636: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3637: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3638: function ConcatSep(const S, S2, Separator: string): string;
3639: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
3640: function ConcatLeftSep(const S, S2, Separator: string): string;
3641: { MinimizeString truncs long string, S, and appends '...' symbols, if Length of S is more than MaxLen }
3642: function MinimizeString(const S: string; const MaxLen: Integer): string;
3643: { Next 4 function for russian chars transliterating.
3644:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes works sucks }
3645: procedure Dos2Win(var S: string);
3646: procedure Win2Dos(var S: string);
3647: function Dos2WinRes(const S: string): string;
3648: function Win2DosRes(const S: string): string;
3649: function Win2Koi(const S: string): string;
3650: { Spaces returns string consists on N space chars }
3651: function Spaces(const N: Integer): string;
3652: { AddSpaces add spaces to string, S, if it Length is smaller than N }
3653: function AddSpaces(const S: string; const N: Integer): string;
3654: { function LastDate for russian users only } { returns date relative to current date: '' }
3655: function LastDate(const Dat: TDateTime): string;
3656: { CurrencyToStr format currency, Cur, using ffCurrency float format}
3657: function CurrencyToStr(const Cur: currency): string;
3658: { Cmp compares two strings and returns True if they are equal. Case-insensitive.}
3659: function Cmp(const S1, S2: string): Boolean;
3660: { StringCat add S2 string to S1 and returns this string }
3661: function StringCat(var S1: string; S2: string): string;
```

```
3662: { HasChar returns True, if Char, Ch, contains in string, S }
3663: function HasChar(const Ch: Char; const S: string): Boolean;
3664: function HasAnyChar(const Chars: string; const S: string): Boolean;
3665: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3666: function CountOfChar(const Ch: Char; const S: string): Integer;
3667: function DefStr(const S: string; Default: string): string;
3668: {**** files routines}
3669: { GetWinDir returns Windows folder name }
3670: function GetWinDir: TFileName;
3671: function GetSysDir: String;
3672: { GetTempDir returns Windows temporary folder name }
3673: function GetTempDir: string;
3674: { GenTempFileName returns temporary file name on drive, there FileName is placed }
3675: function GenTempFileName(FileName: string): string;
3676: { GenTempFileNameExt same to previous function, but returning filename has given extension, FileExt }
3677: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
3678: { ClearDir clears folder Dir }
3679: function ClearDir(const Dir: string): Boolean;
3680: { DeleteDir clears and than delete folder Dir }
3681: function DeleteDir(const Dir: string): Boolean;
3682: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3683: function FileEquMask(FileName, Mask: TFileName): Boolean;
3684: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3685:   Masks must be separated with comma (';') }
3686: function FileEquMasks(FileName, Masks: TFileName): Boolean;
3687: procedure DeleteFiles(const Folder: TFileName; const Masks: string);
3688: { LZFileExpand expand file, FileSource into FileDest.File must be compressed,using MS Compress program }
3689: function LZFileExpand(const FileSource, FileDest: string): Boolean;
3690: { FileGetInfo fills SearchRec record for specified file attributes}
3691: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
3692: { HasSubFolder returns True, if folder APath contains other folders }
3693: function HasSubFolder(APath: TFileName): Boolean;
3694: { IsEmptyFolder returns True, if there are no files or folders in given folder, APath}
3695: function IsEmptyFolder(APath: TFileName): Boolean;
3696: { AddSlash add slash Char to Dir parameter, if needed }
3697: procedure AddSlash(var Dir: TFileName);
3698: { AddSlash returns string with added slash Char to Dir parameter, if needed }
3699: function AddSlash2(const Dir: TFileName): string;
3700: { AddPath returns FileName with Path, if FileName not contain any path }
3701: function AddPath(const FileName, Path: TFileName): TFileName;
3702: function AddPaths(const PathList, Path: string): string;
3703: function ParentPath(const Path: TFileName): TFileName;
3704: function FindInPath(const FileName, PathList: string): TFileName;
3705: function FindInPaths(const fileName,paths: String): String;
3706: {$IFNDEF BCB1}
3707: { BrowseForFolder displays Browse For Folder dialog }
3708: function BrowseForFolder(const Handle: HWND; const Title: string; var Folder: string): Boolean;
3709: {$ENDIF BCB1}
3710: Function BrowseForFolder(const ATitle: string; AllowCreate : Boolean; var ADirectory : string;
     AHelpContext : THelpContext) : Boolean
3711: Function BrowseForComputer(const ATitle : string; AllowCreate : Boolean; var ADirectory : string;
     AHelpContext : THelpContext) : Boolean
3712: Function BrowseDirectory(var AFolderName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3713: Function BrowseComputer(var AComputerName:string;const DlgText:string;AHelpContext:THelpContext):Boolean
3714:
3715: { DeleteReadOnlyFile clears R/O file attribute and delete file }
3716: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
3717: { HasParam returns True, if program running with specified parameter, Param }
3718: function HasParam(const Param: string): Boolean;
3719: function HasSwitch(const Param: string): Boolean;
3720: function Switch(const Param: string): string;
3721: { ExePath returns ExtractFilePath(ParamStr(0)) }
3722: function ExePath: TFileName;
3723: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
3724: function FileTimeToDateTime(const FT: TFileTime): TDateTime;
3725: function MakeValidFileName(const FileName: TFileName; const ReplaceBadChar: Char): TFileName;
3726: {**** Graphic routines }
3727: { TTFontSelected returns True, if True Type font is selected in specified device context }
3728: function TTFontSelected(const DC: HDC): Boolean;
3729: { TrueInflateRect inflates rect in other method, than InflateRect API function }
3730: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
3731: {**** Windows routines }
3732: { SetWindowTop put window to top without recreating window }
3733: procedure SetWindowTop(const Handle: HWND; const Top: Boolean);
3734: {**** other routines }
3735: { KeyPressed returns True, if Key VK is now pressed }
3736: function KeyPressed(VK: Integer): Boolean;
3737: procedure SwapInt(var Int1, Int2: Integer);
3738: function IntPower(Base, Exponent: Integer): Integer;
3739: function ChangeTopException(E: TObject): TObject;
3740: function StrToBool(const S: string): Boolean;
3741: {$IFNDEF COMPILER3_UP}
3742: { AnsiStrLIComp compares S1 to S2, without case-sensitivity, up to a maximum
3743:   Length of MaxLen bytes. The compare operation is controlled by the
3744:   current Windows locale. The return value is the same as for CompareStr. }
3745: function AnsiStrLIComp(S1, S2: PChar; MaxLen: Cardinal): Integer;
3746: function AnsiStrIComp(S1, S2: PChar): Integer;
3747: {$ENDIF}
3748: function Var2Type(V: Variant; const VarType: Integer): Variant;
```

```
3749: function VarToInt(V: Variant): Integer;
3750: function VarToFloat(V: Variant): Double;
3751: { following functions are not documented because they are don't work properly , so don't use them }
3752: function ReplaceSokr1(S: string; const Word, Frase: string): string;
3753: { ReplaceSokr1 is full equal to ReplaceString function - only for compatibility - don't use }
3754: { GetSubStr is full equal to SubStr function - only for compatibility - don't use }
3755: function GetSubStr(const S: string; const Index: Integer; const Separator: Char): string;
3756: function GetParameter: string;
3757: function GetLongFileName(FileName: string): string;
3758: {* from  FileCtrl}
3759: function DirectoryExists(const Name: string): Boolean;
3760: procedure ForceDirectories(Dir: string);
3761: {# from  FileCtrl}
3762: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
3763: function GetComputerID: string;
3764: function GetComputerName: string;
3765: {**** string routines }
3766: { ReplaceAllSokr searches for all substrings, Words,in a string, S, and replaces them with Frases with the
      same Index.Also see RAUtilsW.ReplaceSokr1 function }
3767: function ReplaceAllSokr(S: string; Words, Frases: TStrings): string;
3768: { ReplaceSokr searches the Word in a string, S, on PosBeg position,
3769:   in the list, Words, and if founds, replaces this Word with string from another list, Frases, with the
      same Index, and then update NewSelStart variable }
3770: function ReplaceSokr(S:string;PosBeg,Len:Integer;Words,Frases:TStrings;var NewSelStart:Integer): string;
3771: { CountOfLines calculates the lines count in a string,each line separated from another with CrLf sequence }
3772: function CountOfLines(const S: string): Integer;
3773: { DeleteEmptyLines deletes all empty lines from strings, Ss. Lines contained only spaces also deletes. }
3774: procedure DeleteEmptyLines(Ss: TStrings);
3775: { SQLAddWhere addes or modifies existing where-statement, where, to the strings, SQL.
3776:   Note: If strings SQL allready contains where-statement, it must be started on begining of any line }
3777: procedure SQLAddWhere(SQL: TStrings; const Where: string);
3778: {**** files routines - }
3779: { ResSaveToFile save resource named as Name with Typ type into file FileName.
3780:   Resource can be compressed using MS Compress program}
3781: function ResSaveToFile(const Typ,Name: string; const Compressed:Boolean; const FileName: string): Boolean;
3782: function ResSaveToFileEx(Inst:HINST;Typ,Name:PChar;const Compressed:Bool;const FileName:string): Bool
3783: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
3784: { Execute executes other program and waiting for it terminating, then return its Exit Code }
3785: function ExecuteJ(const CommandLine, WorkingDirectory: string): Integer;
3786: { IniReadSection read section, Section, from ini-file,
3787:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
3788:   Note: TIninFile.ReadSection function reads only strings with '=' symbol.}
3789: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
3790: { LoadTextFile load text file, FileName, into string }
3791: function LoadTextFile(const FileName: TFileName): string;
3792: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3793: { ReadFolder reads files list from disk folder, Folder, that are equal  Mask, into strings, FileList}
3794: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
3795: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
3796: {$IFDEF COMPILER3_UP}
3797: { TargetFileName - if FileName is ShortCut returns filename ShortCut linked to }
3798: function TargetFileName(const FileName: TFileName): TFileName;
3799: { return filename ShortCut linked to }
3800: function ResolveLink(const hWnd: HWND; const LinkFile: TFileName; var FileName: TFileName): HRESULT;
3801: {$ENDIF COMPILER3_UP}
3802: {**** Graphic routines - }
3803: { LoadIcoToImage loads two icons from resource named NameRes,into two image lists ALarge and ASmall}
3804: procedure LoadIcoToImage(ALarge, ASmall: TImageList; const NameRes: string);
3805: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
3806: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
3807: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
3808: function RATextOutEx(Canvas:TCanvas; const R,RClip:TRect;const S: string;const CalcHeight:Boolean):Integer;
3809: { RATextCalcHeight calculate needed height to correct output, using RATextOut or RATextOutEx functions }
3810: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
3811: { Cinema draws some visual effect }
3812: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
3813: { Roughed fills rect with special 3D pattern }
3814: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
3815: { 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 }
3816: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
3817: { TextWidth calculate text with for writing using standard desktop font }
3818: function TextWidth(AStr: string): Integer;
3819: { DefineCursor load cursor from resource, and return available cursor number, assigned to it }
3820: function DefineCursor(Identifer: PChar): TCursor;
3821: {**** other routines - }
3822: { FindFormByClass returns first form specified class, FormClass,owned by Application global variable }
3823: function FindFormByClass(FormClass: TFormClass): TForm;
3824: function FindFormByClassName(FormClassName: string): TForm;
3825: { FindByTag returns the control with specified class, ComponentClass, from WinContol.Controls property,
3826:   having Tag property value, equaled to Tag parameter }
3827: function FindByTag(WinControl:TWinControl;ComponentClass:TComponentClass;const Tag:Integer):TComponent;
3828: { ControlAtPos2 equal to TWinControl.ControlAtPos function, but works better }
3829: function ControlAtPos2(Parent: TWinControl; X, Y: Integer): TControl;
3830: { RBTag searches WinControl.Controls for checked RadioButton and returns its Tag property value }
3831: function RBTag(Parent: TWinControl): Integer;
3832: { AppMinimized returns True, if Application is minimized }
3833: function AppMinimized: Boolean;
3834: { MessageBox is Application.MessageBox with string (not PChar) parameters.
```

```
3835:   if Caption parameter = '', it replaced with Application.Title }
3836: function MessageBoxJ(const Msg: string; Caption: string;const Flags: Integer): Integer;
3837: function MsgDlg2(const Msg, ACaption: string; DlgType: TMsgDlgType;
3838:   Buttons: TMsgDlgButtons; HelpContext: Integer; Control: TWinControl): Integer;
3839: function MsgDlgDef(const Msg, ACaption: string; DlgType: TMsgDlgType;
3840:   Buttons: TMsgDlgButtons; DefButton:TMsgDlgBtn; HelpContext: Integer;Control: TWinControl): Integer;
3841: { Delay stop program execution to MSec msec }
3842: procedure Delay(MSec: Longword);
3843: procedure CenterHor(Parent: TControl; MinLeft: Integer; Controls: array of TControl);
3844: procedure EnableControls(Control: TWinControl; const Enable: Boolean);
3845: procedure EnableMenuItems(MenuItem: TMenuItem; const Tag: Integer; const Enable: Boolean);
3846: procedure ExpandWidth(Parent: TControl; MinWidth: Integer; Controls: array of TControl);
3847: function PanelBorder(Panel: TCustomPanel): Integer;
3848: function Pixels(Control: TControl; APixels: Integer): Integer;
3849: procedure SetChildPropOrd(Owner: TComponent; PropName: string; Value: Longint);
3850: procedure Error(const Msg: string);
3851: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
3852:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
3853:   {ex. Text parameter:'Item 1 <b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green <c:blue>blue</i>'}
3854: function ItemHtDraw(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
3855:   const HideSelColor: Boolean): string;
3856: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;const State: TOwnerDrawState; const Text: string;
3857:   const HideSelColor: Boolean): Integer;
3858: function ItemHtPlain(const Text: string): string;
3859: { ClearList - clears list of TObject }
3860: procedure ClearList(List: TList);
3861: procedure MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
3862: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
3863: { RTTI support }
3864: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
3865: function GetPropStr(Obj: TObject; const PropName: string): string;
3866: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
3867: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
3868: procedure PrepareIniSection(SS: TStrings);
3869: { following functions are not documented because they are don't work properly, so don't use them }
3870: {$IFDEF COMPILER2}
3871: function CompareMem(P1, P2: Pointer; Length: Integer): Boolean; assembler;
3872: {$ENDIF}
3873:
3874: procedure SIRegister_JvBoxProcs(CL: TPSPascalCompiler);
3875: begin
3876:  Procedure BoxMoveSelectedItems( SrcList, DstList : TWinControl)
3877:  Procedure BoxMoveAllItems( SrcList, DstList : TWinControl)
3878:  Procedure BoxDragOver(List:TWinControl;Source:TObject;X,Y:Int;State:TDragState;var
      Accept:Bool;Sorted:Bool;
3879:  Procedure BoxMoveFocusedItem( List : TWinControl; DstIndex : Integer)
3880:  Procedure BoxMoveSelected( List : TWinControl; Items : TStrings)
3881:  Procedure BoxSetItem( List : TWinControl; Index : Integer)
3882:  Function BoxGetFirstSelection( List : TWinControl) : Integer
3883:  Function BoxCanDropItem( List : TWinControl; X, Y : Integer; var DragIndex : Integer) : Boolean
3884: end;
3885:
3886: procedure SIRegister_JvCsvParse(CL: TPSPascalCompiler);
3887: begin
3888:  Const('MaxInitStrNum','LongInt').SetInt( 9);
3889: Function JvAnsiStrSplit( const InString : AnsiString; const SplitChar, QuoteChar:AnsiChar; var OutStrings
      : array of AnsiString; MaxSplit : Integer) : Integer
3890: Function JvStrSplit( const InString : string; const SplitChar, QuoteChar : Char; var OutStrings : array of
      string; MaxSplit : Integer) : Integer
3891: Function JvAnsiStrSplitStrings(const InStr:AnsiString;const SplitChar,
      QuoteChar:AnsiChar;OutStrs:TStrings):Integer;
3892: Function JvAnsiStrStrip( S : AnsiString) : AnsiString
3893: Function JvStrStrip( S : string) : string
3894: Function GetString( var Source : AnsiString; const Separator : AnsiString) : AnsiString
3895: Function PadString( const S : AnsiString; Len : Integer; PadChar : AnsiChar) : AnsiString
3896: Function BuildPathName( const PathName, FileName : AnsiString) : AnsiString
3897: Function StrEatWhiteSpace( const S : string) : string
3898: Function HexToAscii( const S : AnsiString) : AnsiString
3899: Function AsciiToHex( const S : AnsiString) : AnsiString
3900: Function StripQuotes( const S1 : AnsiString) : AnsiString
3901: Function ValidNumericLiteral( S1 : PAnsiChar) : Boolean
3902: Function ValidIntLiteral( S1 : PAnsiChar) : Boolean
3903: Function ValidHexLiteral( S1 : PAnsiChar) : Boolean
3904: Function HexPCharToInt( S1 : PAnsiChar) : Integer
3905: Function ValidStringLiteral( S1 : PAnsiChar) : Boolean
3906: Function StripPCharQuotes( S1 : PAnsiChar) : AnsiString
3907: Function JvValidIdentifierAnsi( S1 : PAnsiChar) : Boolean
3908: Function JvValidIdentifier( S1 : String) : Boolean
3909: Function JvEndChar( X : AnsiChar) : Boolean
3910: Procedure JvGetToken( S1, S2 : PAnsiChar)
3911: Function IsExpressionKeyword( S1 : PAnsiChar) : Boolean
3912: Function IsKeyword( S1 : PAnsiChar) : Boolean
3913: Function JvValidVarReference( S1 : PAnsiChar) : Boolean
3914: Function GetParenthesis( S1, S2 : PAnsiChar) : Boolean
3915: Procedure JvGetVarReference( S1, S2, SIdx : PAnsiChar)
3916: Procedure JvEatWhitespaceChars( S1 : PAnsiChar);
3917: Procedure JvEatWhitespaceChars1( S1 : PWideChar);
3918: Function GetTokenCount : Integer
3919: Procedure ResetTokenCount
```

```
3920: end;
3921:
3922: procedure SIRegister_JvDBQueryParamsForm(CL: TPSPascalCompiler);
3923: begin
3924:   SIRegister_TJvQueryParamsDialog(CL);
3925:  Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext) : Boolean
3926: end;
3927:
3928: ******************************** JvStrUtil /  JvStrUtils;****************************
3929: function FindNotBlankCharPos(const S: string): Integer;
3930: function AnsiChangeCase(const S: string): string;
3931: function GetWordOnPos(const S: string; const P: Integer): string;
3932: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3933: function Cmp(const S1, S2: string): Boolean;
3934: { Spaces returns string consists on N space chars }
3935: function Spaces(const N: Integer): string;
3936: { HasChar returns True, if char, Ch, contains in string, S }
3937: function HasChar(const Ch: Char; const S: string): Boolean;
3938: function HasAnyChar(const Chars: string; const S: string): Boolean;
3939: { SubStr returns substring from string, S, separated with Separator string}
3940: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3941: { SubStrEnd same to previous function but Index numerated from the end of string }
3942: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3943: { ReplaceString searches for all substrings, OldPattern, in a string, S, replaces them with NewPattern }
3944: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3945: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3946: { GetXYByPos is same to previous function, but returns X position in line too}
3947: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3948: { AddSlash returns string with added slash char to Dir parameter, if needed }
3949: function AddSlash2(const Dir: TFileName): string;
3950: { AddPath returns FileName with Path, if FileName not contain any path }
3951: function AddPath(const FileName, Path: TFileName): TFileName;
3952: { ExePath returns ExtractFilePath(ParamStr(0)) }
3953: function ExePath: TFileName;
3954: function LoadTextFile(const FileName: TFileName): string;
3955: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3956: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3957: function ConcatSep(const S, S2, Separator: string): string;
3958: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3959: function FileEquMask(FileName, Mask: TFileName): Boolean;
3960: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3961:   Masks must be separated with comma (';') }
3962: function FileEquMasks(FileName, Masks: TFileName): Boolean;
3963: function StringEndsWith(const Str, SubStr: string): Boolean;
3964: function ExtractFilePath2(const FileName: string): string;
3965: function StrToOem(const AnsiStr: string): string;
3966: { StrToOem translates a string from the Windows character set into the OEM character set. }
3967: function OemToAnsiStr(const OemStr: string): string;
3968: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
3969: function IsEmptyStr(const S: string; const EmptyChars: TCharSet): Boolean;
3970: { EmptyStr returns true if the given string contains only character from the EmptyChars. }
3971: function ReplaceStr(const S, Srch, Replace: string): string;
3972: { Returns string with every occurrence of Srch string replaced with Replace string. }
3973: function DelSpace(const S: string): string;
3974: { DelSpace return a string with all white spaces removed. }
3975: function DelChars(const S: string; Chr: Char): string;
3976: { DelChars return a string with all Chr characters removed. }
3977: function DelBSpace(const S: string): string;
3978: { DelBSpace trims leading spaces from the given string. }
3979: function DelESpace(const S: string): string;
3980: { DelESpace trims trailing spaces from the given string. }
3981: function DelRSpace(const S: string): string;
3982: { DelRSpace trims leading and trailing spaces from the given string. }
3983: function DelSpace1(const S: string): string;
3984: { DelSpace1 return a string with all non-single white spaces removed. }
3985: function Tab2Space(const S: string; Numb: Byte): string;
3986: { Tab2Space converts any tabulation character in the given string to the Numb spaces characters. }
3987: function NPos(const C: string; S: string; N: Integer): Integer;
3988: { NPos searches for a N-th position of substring C in a given string. }
3989: function MakeStr(C: Char; N: Integer): string;
3990: function MS(C: Char; N: Integer): string;
3991: { MakeStr return a string of length N filled with character C. }
3992: function AddChar(C: Char; const S: string; N: Integer): string;
3993: { AddChar return a string left-padded to length N with characters C. }
3994: function AddCharR(C: Char; const S: string; N: Integer): string;
3995: { AddCharR return a string right-padded to length N with characters C. }
3996: function LeftStr(const S: string; N: Integer): string;
3997: { LeftStr return a string right-padded to length N with blanks. }
3998: function RightStr(const S: string; N: Integer): string;
3999: { RightStr return a string left-padded to length N with blanks. }
4000: function CenterStr(const S: string; Len: Integer): string;
4001: { CenterStr centers the characters in the string based upon the Len specified. }
4002: function CompStr(const S1, S2: string): Integer;
4003: {CompStr compares S1 to S2, case-sensitivity. return val is -1 if S1 < S2,0 if S1 = S2,or 1 if S1>S2. }
4004: function CompText(const S1, S2: string): Integer;
4005: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4006: function Copy2Symb(const S: string; Symb: Char): string;
4007: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4008: function Copy2SymbDel(var S: string; Symb: Char): string;
```

```
4009: { Copy2SymbDel returns a substr of string S from to first character Symb and removes substring from S. }
4010: function Copy2Space(const S: string): string;
4011: { Copy2Symb returns a substring of a string S from begining to first white space. }
4012: function Copy2SpaceDel(var S: string): string;
4013: { Copy2SpaceDel returns a substring of a string S from begining to first
4014:   white space and removes this substring from S. }
4015: function AnsiProperCase(const S: string; const WordDelims: TCharSet): string;
4016: { Returns string, with the first letter of each word in uppercase,
4017:   all other letters in lowercase. Words are delimited by WordDelims. }
4018: function WordCount(const S: string; const WordDelims: TCharSet): Integer;
4019: { WordCount given a set of word delimiters, returns number of words in S. }
4020: function WordPosition(const N: Integer; const S: string; const WordDelims: TCharSet): Integer;
4021: { Given a set of word delimiters, returns start position of N'th word in S. }
4022: function ExtractWord(N: Integer; const S: string; const WordDelims: TCharSet): string;
4023: function ExtractWordPos(N:Integer; const S:string; const WordDelims:TCharSet;var Pos: Integer): string;
4024: function ExtractDelimited(N: Integer; const S: string; const Delims: TCharSet): string;
4025: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4026:   delimiters, return the N'th word in S. }
4027: function ExtractSubstr(const S: string; var Pos: Integer; const Delims: TCharSet): string;
4028: { ExtractSubstr given set of word delimiters,returns the substring from S,that started from position Pos.}
4029: function IsWordPresent(const W, S: string; const WordDelims: TCharSet): Boolean;
4030: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4031: function QuotedString(const S: string; Quote: Char): string;
4032: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4033: function ExtractQuotedString(const S: string; Quote: Char): string;
4034: { ExtractQuotedString removes the Quote characters from the beginning and end of a quoted string,
4035:    and reduces pairs of Quote characters within the quoted string to a single character. }
4036: function FindPart(const HelpWilds, InputStr: string): Integer;
4037: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4038: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4039: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4040: function XorString(const Key, Src: ShortString): ShortString;
4041: function XorEncode(const Key, Source: string): string;
4042: function XorDecode(const Key, Source: string): string;
4043: { ** Command line routines ** }
4044: {$IFNDEF COMPILER4_UP}
4045: function FindCmdLineSwitch(const Switch: string; SwitchChars: TCharSet;IgnoreCase: Boolean): Boolean;
4046: {$ENDIF}
4047: function GetCmdLineArg(const Switch: string; SwitchChars: TCharSet): string;
4048: { ** Numeric string handling routines ** }
4049: function Numb2USA(const S: string): string;
4050: { Numb2USA converts numeric string S to USA-format. }
4051: function Dec2Hex(N: Longint; A: Byte): string;
4052: function D2H(N: Longint; A: Byte): string;
4053: { Dec2Hex converts the given value to a hexadecimal string representation
4054:   with the minimum number of digits (A) specified. }
4055: function Hex2Dec(const S: string): Longint;
4056: function H2D(const S: string): Longint;
4057: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4058: function Dec2Numb(N: Longint; A, B: Byte): string;
4059: { Dec2Numb converts the given value to a string representation with the
4060:   base equal to B and with the minimum number of digits (A) specified. }
4061: function Numb2Dec(S: string; B: Byte): Longint;
4062: { Numb2Dec converts the given B-based numeric string to the corresponding integer value. }
4063: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4064: { IntToBin converts given value to a bin string representation with min number of digits specified. }
4065: function IntToRoman(Value: Longint): string;
4066: { IntToRoman converts the given value to a roman numeric string representation. }
4067: function RomanToInt(const S: string): Longint;
4068: { RomanToInt converts the given string to an integer value. If the string
4069:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4070: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
4071: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
4072: ********************************* JvFileUtil;*********************************
4073: procedure CopyFileJ(const FileName, DestName: string; ProgressControl: TControl);
4074: procedure CopyFileEx(const FileName,DestName:string;OverwriteReadOnly,ShellDialog:Boolean;ProgressControl:
     TControl);
4075: procedure MoveFile(const FileName, DestName: TFileName);
4076: procedure MoveFileEx(const FileName, DestName: TFileName; ShellDialog: Boolean);
4077: {$IFDEF COMPILER4_UP}
4078: function GetFileSize(const FileName: string): Int64;
4079: {$ELSE}
4080: function GetFileSize(const FileName: string): Longint;
4081: {$ENDIF}
4082: function FileDateTime(const FileName: string): TDateTime;
4083: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4084: function DeleteFiles(const FileMask: string): Boolean;
4085: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4086: function ClearDir(const Path: string; Delete: Boolean): Boolean;
4087: function NormalDir(const DirName: string): string;
4088: function RemoveBackSlash(const DirName: string): string;
4089: function ValidFileName(const FileName: string): Boolean;
4090: function DirExists(Name: string): Boolean;
4091: procedure ForceDirectories(Dir: string);
4092: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer;
4093:   {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4094: {$IFDEF COMPILER4_UP}
4095: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4096: {$ENDIF}
```

```
4097: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer;
4098: {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4099: {$IFDEF COMPILER4_UP}
4100: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4101: {$ENDIF}
4102: function GetTempDir: string;
4103: function GetWindowsDir: string;
4104: function GetSystemDir: string;
4105: function BrowseDirectory(var AFolderName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4106: {$IFDEF WIN32}
4107: function BrowseComputer(var ComputerName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4108: function ShortToLongFileName(const ShortName: string): string;
4109: function ShortToLongPath(const ShortName: string): string;
4110: function LongToShortFileName(const LongName: string): string;
4111: function LongToShortPath(const LongName: string): string;
4112: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4113: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4114: {$ENDIF WIN32}
4115: {$IFNDEF COMPILER3_UP}
4116: function IsPathDelimiter(const S: string; Index: Integer): Boolean;
4117: {$ENDIF}
4118: Function CreateCalculatorForm( AOwner : TComponent; AHelpContext : THelpContext) : TJvCalculatorForm
4119: Function CreatePopupCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode) : TWinControl
4120: Function CreatePopupCalculator( AOwner : TComponent) : TWinControl
4121: Procedure SetupPopupCalculator( PopupCalc : TWinControl; APrecision : Byte; ABeepOnError : Boolean)
4122:
4123: //**************************procedure SIRegister_VarHlpr(CL: TPSPascalCompiler);
4124:  Procedure VariantClear( var V : Variant)
4125:  Procedure VariantArrayRedim( var V : Variant; High : Integer)
4126:  Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
4127:  Procedure VariantCpy( const src : Variant; var dst : Variant)
4128:  Procedure VariantAdd( const src : Variant; var dst : Variant)
4129:  Procedure VariantSub( const src : Variant; var dst : Variant)
4130:  Procedure VariantMul( const src : Variant; var dst : Variant)
4131:  Procedure VariantDiv( const src : Variant; var dst : Variant)
4132:  Procedure VariantMod( const src : Variant; var dst : Variant)
4133:  Procedure VariantAnd( const src : Variant; var dst : Variant)
4134:  Procedure VariantOr( const src : Variant; var dst : Variant)
4135:  Procedure VariantXor( const src : Variant; var dst : Variant)
4136:  Procedure VariantShl( const src : Variant; var dst : Variant)
4137:  Procedure VariantShr( const src : Variant; var dst : Variant)
4138:  Function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
4139:  Function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
4140:  Function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
4141:  Function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
4142:  Function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
4143:  Function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
4144:  Function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
4145:  Function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
4146:  Function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
4147:  Function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
4148:  Function VariantNot( const V1 : Variant) : Variant
4149:  Function VariantNeg( const V1 : Variant) : Variant
4150:  Function VariantGetElement( const V : Variant; i1 : integer) : Variant;
4151:  Function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
4152:  Function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
4153:  Function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
4154:  Function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
4155:  Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
4156:  Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
4157:  Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
4158:  Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
4159:  Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
4160: end;
4161:
4162: ************************************unit uPSI_JvgUtils;************************************
4163: function IsEven(I: Integer): Boolean;
4164: function InchesToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4165: function CentimetersToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4166: procedure SwapInt(var I1, I2: Integer);
4167: function Spaces(Count: Integer): string;
4168: function DupStr(const Str: string; Count: Integer): string;
4169: function DupChar(C: Char; Count: Integer): string;
4170: procedure Msg(const AMsg: string);
4171: function RectW(R: TRect): Integer;
4172: function RectH(R: TRect): Integer;
4173: function IncColor(AColor: Longint; AOffset: Byte): Longint;
4174: function DecColor(AColor: Longint; AOffset: Byte): Longint;
4175: function IsItAFilledBitmap(Bmp: TBitmap): Boolean;
4176: procedure DrawTextInRectWithAlign(DC: HDC; R: TRect; const Text: string;
4177:   HAlign: TglHorAlign; VAlign: TglVertAlign; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
4178: procedure DrawTextInRect(DC: HDC; R:TRect; const Text:string;Style:TglTextStyle;Fnt:TFont;Flags: UINT);
4179: procedure ExtTextOutExt(DC: HDC; X, Y: Integer; R: TRect; const Text: string;
4180:   Style: TglTextStyle; ADelineated, ASupress3D: Boolean; FontColor, DelinColor,HighlightColor,ShadowColor:
      TColor; Illumination: TJvgIllumination; Gradient: TJvgGradient; Font: TFont);
4181: procedure DrawBox(DC:HDC; var R:TRect; Style:TglBoxStyle;BackgrColor: Longint; ATransparent: Boolean);
4182: function DrawBoxEx(DC: HDC; ARect: TRect; Borders: TglSides;
4183:   BevelInner, BevelOuter: TPanelBevel; Bold:Boolean; BackgrColor: Longint;ATransparent: Boolean): TRect;
4184: procedure GradientBox(DC: HDC; R: TRect; Gradient: TJvgGradient;PenStyle, PenWidth: Integer);
```

```
4185: procedure ChangeBitmapColor(Bitmap: TBitmap; FromColor, ToColor: TColor);
4186: procedure DrawBitmapExt(DC: HDC; { DC - background & result}
4187:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
4188:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4189:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4190: procedure CreateBitmapExt(DC: HDC; { DC - background & result}
4191:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; //...X,Y _in_ rect!
4192:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4193:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4194: procedure BringParentWindowToTop(Wnd: TWinControl);
4195: function GetParentForm(Control: TControl): TForm;
4196: procedure GetWindowImageFrom(Control:TWinControl;X,Y:Integer;ADrawSelf,ADrawChildWindows:Boolean;DC:HDC)
4197: procedure GetWindowImage(Control: TWinControl; ADrawSelf, ADrawChildWindows: Boolean; DC: HDC);
4198: procedure GetParentImageRect(Control: TControl; Rect: TRect; DC: HDC);
4199: function CreateRotatedFont(F: TFont; Escapement: Integer): HFONT;
4200: function FindMainWindow(const AWndClass, AWndTitle: string): THandle;
4201: procedure CalcShadowAndHighlightColors(BaseColor: TColor; Colors: TJvgLabelColors);
4202: function CalcMathString(AExpression: string): Single;
4203: function IIF(AExpression: Boolean; IfTrue, IfFalse: Variant): Variant; overload;
4204: function IIF(AExpression: Boolean; const IfTrue, IfFalse: string): string; overload;
4205: function GetTransparentColor(Bitmap: TBitmap; AutoTrColor: TglAutoTransparentColor): TColor;
4206: procedure TypeStringOnKeyboard(const S: string);
4207: function NextStringGridCell( Grid: TStringGrid ): Boolean;
4208: procedure DrawTextExtAligned(Canvas:TCanvas;const
     Text:string;R:TRect;Alignment:TglAlignment;WordWrap:Boolean);
4209: procedure LoadComponentFromTextFile(Component: TComponent; const FileName: string);
4210: procedure SaveComponentToTextFile(Component: TComponent; const FileName: string);
4211: function ComponentToString(Component: TComponent): string;
4212: procedure StringToComponent(Component: TComponent; const Value: string);
4213: function PlayWaveResource(const ResName: string): Boolean;
4214: function UserName: string;
4215: function ComputerName: string;
4216: function CreateIniFileName: string;
4217: function ExpandString(const Str: string; Len: Integer): string;
4218: function Transliterate(const Str: string; RusToLat: Boolean): string;
4219: function IsSmallFonts: Boolean;
4220: function SystemColorDepth: Integer;
4221: function GetFileTypeJ(const FileName: string): TglFileType;
4222: function FindControlAtPt(Control: TWinControl; Pt: TPoint; MinClass: TClass): TControl;
4223: function StrPosExt(const Str1, Str2: PChar; Str2Len: DWORD): PChar;
4224:
4225: { ************************************Utility routines of unit classes}
4226: function LineStart(Buffer, BufPos: PChar): PChar
4227: function ExtractStrings(Separators, WhiteSpace: TSysCharSet; Content: PChar;'+
4228:   'Strings: TStrings): Integer
4229:  Function TestStreamFormat( Stream : TStream) : TStreamOriginalFormat
4230:  Procedure RegisterClass( AClass : TPersistentClass)
4231:  Procedure RegisterClasses( AClasses : array of TPersistentClass)
4232:  Procedure RegisterClassAlias( AClass : TPersistentClass; const Alias : string)
4233:  Procedure UnRegisterClass( AClass : TPersistentClass)
4234:  Procedure UnRegisterClasses( AClasses : array of TPersistentClass)
4235:  Procedure UnRegisterModuleClasses( Module : HMODULE)
4236:  Function FindGlobalComponent( const Name : string) : TComponent
4237:  Function IsUniqueGlobalComponentName( const Name : string) : Boolean
4238:  Function InitInheritedComponent( Instance : TComponent; RootAncestor : TClass) : Boolean
4239:  Function InitComponentRes( const ResName : string; Instance : TComponent) : Boolean
4240:  Function ReadComponentRes( const ResName : string; Instance : TComponent) : TComponent
4241:  Function ReadComponentResEx( HInstance : THandle; const ResName : string) : TComponent
4242:  Function ReadComponentResFile( const FileName : string; Instance : TComponent) : TComponent
4243:  Procedure WriteComponentResFile( const FileName : string; Instance : TComponent)
4244:  Procedure GlobalFixupReferences
4245:  Procedure GetFixupReferenceNames( Root : TComponent; Names : TStrings)
4246:  Procedure GetFixupInstanceNames(Root: TComponent; const ReferenceRootName string; Names : TStrings)
4247:  Procedure RedirectFixupReferences( Root : TComponent; const OldRootName, NewRootName : string)
4248:  Procedure RemoveFixupReferences( Root : TComponent; const RootName : string)
4249:  Procedure RemoveFixups( Instance : TPersistent)
4250:  Function FindNestedComponent( Root : TComponent; const NamePath : string) : TComponent
4251:  Procedure BeginGlobalLoading
4252:  Procedure NotifyGlobalLoading
4253:  Procedure EndGlobalLoading
4254:  Function GetUltimateOwner1( ACollection : TCollection) : TPersistent;
4255:  Function GetUltimateOwner( APersistent : TPersistent) : TPersistent;
4256:  // AddTypeS('TWndMethod', 'Procedure (var Message : TMessage)
4257:  //Function MakeObjectInstance( Method : TWndMethod) : Pointer
4258:  Procedure FreeObjectInstance( ObjectInstance : Pointer)
4259: // Function AllocateHWnd( Method : TWndMethod) : HWND
4260:  Procedure DeallocateHWnd( Wnd : HWND)
4261:  Function AncestorIsValid( Ancestor : TPersistent; Root, RootAncestor : TComponent) : Boolean
4262: ************************************unit uPSI_SqlTimSt and DB;************************************
4263: Procedure VarSQLTimeStampCreate4( var aDest : Variant; const ASQLTimeStamp : TSQLTimeStamp);
4264: Function VarSQLTimeStampCreate3: Variant;
4265: Function VarSQLTimeStampCreate2( const AValue : string) : Variant;
4266: Function VarSQLTimeStampCreate1( const AValue : TDateTime) : Variant;
4267: Function VarSQLTimeStampCreate( const ASQLTimeStamp : TSQLTimeStamp) : Variant;
4268: Function VarSQLTimeStamp : TVarType
4269: Function VarIsSQLTimeStamp( const aValue : Variant) : Boolean;
4270: Function LocalToUTC( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp //beta
4271: Function UTCToLocal( var TZInfo : TTimeZone; var Value : TSQLTimeStamp) : TSQLTimeStamp //beta
4272: Function VarToSQLTimeStamp( const aValue : Variant) : TSQLTimeStamp
```

```
4273: Function SQLTimeStampToStr( const Format : string; DateTime : TSQLTimeStamp) : string
4274: Function SQLDayOfWeek( const DateTime : TSQLTimeStamp) : integer
4275: Function DateTimeToSQLTimeStamp( const DateTime : TDateTime) : TSQLTimeStamp
4276: Function SQLTimeStampToDateTime( const DateTime : TSQLTimeStamp) : TDateTime
4277: Function TryStrToSQLTimeStamp( const S : string; var TimeStamp : TSQLTimeStamp) : Boolean
4278: Function StrToSQLTimeStamp( const S : string) : TSQLTimeStamp
4279: Procedure CheckSqlTimeStamp( const ASQLTimeStamp : TSQLTimeStamp)
4280: Function ExtractFieldName( const Fields : string; var Pos : Integer) : string;
4281: Function ExtractFieldName( const Fields : WideString; var Pos : Integer) : WideString;
4282:  //'Procedure RegisterFields( const FieldClasses : array of TFieldClass)
4283: Procedure DatabaseError( const Message : WideString; Component : TComponent)
4284: Procedure DatabaseErrorFmt(const Message:WIdeString; const Args:array of const;Component:TComponent)
4285: Procedure DisposeMem( var Buffer, Size : Integer)
4286: Function BuffersEqual( Buf1, Buf2 : Pointer; Size : Integer) : Boolean
4287: Function GetFieldProperty(DataSet:TDataSet; Control:TComponent; const FieldName: WideString): TField
4288: Function VarTypeToDataType( VarType : Integer) : TFieldType
4289: ********************************************unit JvStrings;**************************************
4290: {template functions}
4291: function ReplaceFirst(const SourceStr, FindStr, ReplaceStr: string): string;
4292: function ReplaceLast(const SourceStr, FindStr, ReplaceStr: string): string;
4293: function InsertLastBlock(var SourceStr: string; BlockStr: string): Boolean;
4294: function RemoveMasterBlocks(const SourceStr: string): string;
4295: function RemoveFields(const SourceStr: string): string;
4296: {http functions}
4297: function URLEncode(const Value: AnsiString): AnsiString; // Converts string To A URLEncoded string
4298: function URLDecode(const Value: AnsiString): AnsiString; // Converts string From A URLEncoded string
4299: {set functions}
4300: procedure SplitSet(AText: string; AList: TStringList);
4301: function JoinSet(AList: TStringList): string;
4302: function FirstOfSet(const AText: string): string;
4303: function LastOfSet(const AText: string): string;
4304: function CountOfSet(const AText: string): Integer;
4305: function SetRotateRight(const AText: string): string;
4306: function SetRotateLeft(const AText: string): string;
4307: function SetPick(const AText: string; AIndex: Integer): string;
4308: function SetSort(const AText: string): string;
4309: function SetUnion(const Set1, Set2: string): string;
4310: function SetIntersect(const Set1, Set2: string): string;
4311: function SetExclude(const Set1, Set2: string): string;
4312: {replace any <,> etc by &lt; &gt;}
4313: function XMLSafe(const AText: string): string;
4314: {simple hash, Result can be used in Encrypt}
4315: function Hash(const AText: string): Integer;
4316: { Base64 encode and decode a string }
4317: function B64Encode(const S: AnsiString): AnsiString;
4318: function B64Decode(const S: AnsiString): AnsiString;
4319: {Basic encryption from a Borland Example}
4320: function Encrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4321: function Decrypt(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4322: {Using Encrypt and Decrypt in combination with B64Encode and B64Decode}
4323: function EncryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4324: function DecryptB64(const InString: AnsiString; StartKey, MultKey, AddKey: Integer): AnsiString;
4325: procedure CSVToTags(Src, Dst: TStringList);
4326: // converts a csv list to a tagged string list
4327: procedure TagsToCSV(Src, Dst: TStringList);
4328: // converts a tagged string list to a csv list
4329: // only fieldnames from the first record are scanned ib the other records
4330: procedure ListSelect(Src, Dst: TStringList; const AKey, AValue: string);
4331: {selects akey=avalue from Src and returns recordset in Dst}
4332: procedure ListFilter(Src: TStringList; const AKey, AValue: string);
4333: {filters Src for akey=avalue}
4334: procedure ListOrderBy(Src: TStringList; const AKey: string; Numeric: Boolean);
4335: {orders a tagged Src list by akey}
4336: function PosStr(const FindString, SourceString: string;
4337:   StartPos: Integer = 1): Integer;
4338: { PosStr searches the first occurrence of a substring FindString in a string
4339:   given by SourceString with case sensitivity (upper and lower case characters
4340:   are differed). This function returns the index value of the first character
4341:   of a specified substring from which it occurs in a given string starting with
4342:   StartPos character index. If a specified substring is not found Q_PosStr
4343:   returns zero. author of algorithm is Peter Morris (UK) (Faststrings unit from www.torry.ru). }
4344: function PosStrLast(const FindString, SourceString: string): Integer;
4345: {finds the last occurance}
4346: function LastPosChar(const FindChar: Char; SourceString: string): Integer;
4347: function PosText(const FindString, SourceString: string; StartPos: Integer = 1): Integer;
4348: { PosText searches the first occurrence of a substring FindString in a string
4349:   given by SourceString without case sensitivity (upper and lower case characters are not differed). This
     function returns the index value of the first character of a specified substring from which it occurs in a
     given string starting with Start
4350: function PosTextLast(const FindString, SourceString: string): Integer;
4351: {finds the last occurance}
4352: function NameValuesToXML(const AText: string): string;
4353: {$IFDEF MSWINDOWS}
4354: procedure LoadResourceFile(AFile: string; MemStream: TMemoryStream);
4355: {$ENDIF MSWINDOWS}
4356: procedure DirFiles(const ADir, AMask: string; AFileList: TStringList);
4357: procedure RecurseDirFiles(const ADir: string; var AFileList: TStringList);
4358: procedure RecurseDirProgs(const ADir: string; var AFileList: TStringList);
4359: procedure SaveString(const AFile, AText: string);
```

```
4360: Procedure SaveStringasFile( const AFile, AText : string)
4361: function LoadStringJ(const AFile: string): string;
4362: Function LoadStringofFile( const AFile : string) : string
4363: Procedure SaveStringtoFile( const AFile, AText : string)
4364: Function LoadStringfromFile( const AFile : string) : string
4365: function HexToColor(const AText: string): TColor;
4366: function UppercaseHTMLTags(const AText: string): string;
4367: function LowercaseHTMLTags(const AText: string): string;
4368: procedure GetHTMLAnchors(const AFile: string; AList: TStringList);
4369: function RelativePath(const ASrc, ADst: string): string;
4370: function GetToken(var Start: Integer; const SourceText: string): string;
4371: function PosNonSpace(Start: Integer; const SourceText: string): Integer;
4372: function PosEscaped(Start: Integer; const SourceText, FindText: string; EscapeChar: Char): Integer;
4373: function DeleteEscaped(const SourceText: string; EscapeChar: Char): string;
4374: function BeginOfAttribute(Start: Integer; const SourceText: string): Integer;
4375: // parses the beginning of an attribute: space + alpha character
4376: function ParseAttribute(var Start:Integer; const SourceText:string; var AName,AValue:string): Boolean;
4377: //parses a name="value" attrib from Start; returns 0 when not found or else the position behind attribute
4378: procedure ParseAttributes(const SourceText: string; Attributes: TStrings);
4379: // parses all name=value attributes to the attributes TStringList
4380: function HasStrValue(const AText, AName: string; var AValue: string): Boolean;
4381: // checks if a name="value" pair exists and returns any value
4382: function GetStrValue(const AText, AName, ADefault: string): string;
4383: // retrieves string value from a line like:
4384: //  name="jan verhoeven" email="jan1 dott verhoeven att wxs dott nl"
4385: // returns ADefault when not found
4386: function GetHTMLColorValue(const AText, AName: string; ADefault: TColor): TColor;
4387: // same for a color
4388: function GetIntValue(const AText, AName: string; ADefault: Integer): Integer;
4389: // same for an Integer
4390: function GetFloatValue(const AText, AName: string; ADefault: Extended): Extended;
4391: // same for a float
4392: function GetBoolValue(const AText, AName: string): Boolean;
4393: // same for Boolean but without default
4394: function GetValue(const AText, AName: string): string;
4395: //retrieves string value from a line like: name="jan verhoeven" email="jan1 verhoeven att wxs dott nl"
4396: procedure SetValue(var AText: string; const AName, AValue: string);
4397: // sets a string value in a line
4398: procedure DeleteValue(var AText: string; const AName: string);
4399: // deletes a AName="value" pair from AText
4400: procedure GetNames(AText: string; AList: TStringList);
4401: // get a list of names from a string with name="value" pairs
4402: function GetHTMLColor(AColor: TColor): string;
4403: // converts a color value to the HTML hex value
4404: function BackPosStr(Start: Integer; const FindString, SourceString: string): Integer;
4405: // finds a string backward case sensitive
4406: function BackPosText(Start: Integer; const FindString, SourceString: string): Integer;
4407: // finds a string backward case insensitive
4408: function PosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4409:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4410: // finds a text range, e.g. <TD>....</TD> case sensitive
4411: function PosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4412:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4413: // finds a text range, e.g. <TD>....</td> case insensitive
4414: function BackPosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4415:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4416: // finds a text range backward, e.g. <TD>....</TD> case sensitive
4417: function BackPosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4418:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4419: // finds a text range backward, e.g. <TD>....</td> case insensitive
4420: function PosTag(Start: Integer; SourceString: string; var RangeBegin: Integer;
4421:   var RangeEnd: Integer): Boolean;
4422: // finds a HTML or XML tag:  <....>
4423: function InnerTag(Start: Integer; const HeadString, TailString, SourceString: string;
4424:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4425: // finds the innertext between opening and closing tags
4426: function Easter(NYear: Integer): TDateTime;
4427: // returns the easter date of a year.
4428: function GetWeekNumber(Today: TDateTime): string;
4429: //gets a datecode. Returns year and weeknumber in format: YYWW
4430: function ParseNumber(const S: string): Integer;
4431: // parse number returns the last position, starting from 1
4432: function ParseDate(const S: string): Integer;
4433: // parse a SQL style data string from positions 1,
4434: // starts and ends with #
4435:
4436: ****************************************unit JvJCLUtils;****************************************
4437:
4438: function VarIsInt(Value: Variant): Boolean;
4439:  // VarIsInt returns VarIsOrdinal-[varBoolean]
4440: { PosIdx returns the index of the first appearance of SubStr in Str. The search starts at index "Index". }
4441: function PosIdx(const SubStr, S: string; Index: Integer = 0): Integer;
4442: function PosIdxW(const SubStr, S: WideString; Index: Integer = 0): Integer;
4443: function PosLastCharIdx(Ch: Char; const S: string; Index: Integer = 0): Integer;
4444: { GetWordOnPos returns Word from string, S, on the cursor position, P}
4445: function GetWordOnPos(const S: string; const P: Integer): string;
4446: function GetWordOnPosW(const S: WideString; const P: Integer): WideString;
4447: function GetWordOnPos2(const S: string; P: Integer; var iBeg, iEnd: Integer): string;
4448: function GetWordOnPos2W(const S: WideString; P: Integer; var iBeg, iEnd: Integer): WideString;
```

```
4449: { GetWordOnPosEx working like GetWordOnPos function, but
4450:   also returns Word position in iBeg, iEnd variables }
4451: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4452: function GetWordOnPosExW(const S: WideString; const P: Integer; var iBeg, iEnd: Integer): WideString;
4453: function GetNextWordPosEx(const Text: string; StartIndex: Integer; var iBeg, iEnd: Integer): string;
4454: function GetNextWordPosExW(const Text:WideString;StartIndex:Integer; var iBeg,iEnd:Integer):WideString;
4455: procedure GetEndPosCaret(const Text: string; CaretX, CaretY: Integer; var X, Y: Integer);
4456: { GetEndPosCaret returns the caret position of the last char. For the position
4457:   after the last char of Text you must add 1 to the returned X value. }
4458: procedure GetEndPosCaretW(const Text: WideString;CaretX,CaretY:Integer;var X,Y:Integer);
4459: { GetEndPosCaret returns the caret position of the last char. For the position
4460:   after the last char of Text you must add 1 to the returned X value. }
4461: { SubStrBySeparator returns substring from string, S, separated with Separator string}
4462: function SubStrBySeparator(const S:string;const Index:Integer;const
      Separator:string;StartIndex:Int=1):string;
4463: function SubStrBySeparatorW(const S:WideString;const Index:Int;const
      Separator:WideString;StartIndex:Int:WideString;
4464: { SubStrEnd same to previous function but Index numerated from the end of string }
4465: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4466: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
4467: function SubWord(P: PChar; var P2: PChar): string;
4468: function CurrencyByWord(Value: Currency): string;
4469: { GetLineByPos returns the Line number, there the symbol Pos is pointed. Lines separated with #13 symbol }
4470: function GetLineByPos(const S: string; const Pos: Integer): Integer;
4471: { GetXYByPos is same as GetLineByPos, but returns X position in line as well}
4472: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4473: procedure GetXYByPosW(const S: WideString; const Pos: Integer; var X, Y: Integer);
4474: { ReplaceString searches for all substrings, OldPattern,
4475:   in a string, S, and replaces them with NewPattern }
4476: function ReplaceString(S: string; const OldPattern,NewPattern: string; StartIndex:Integer = 1):string;
4477: function ReplaceStringW(S: WideString; const OldPattern,NewPattern:
      WideString;StartIndex:Integer=1):WideString;
4478: { ConcatSep concatenate S1 and S2 strings with Separator. if S = '' then separator not included }
4479: function ConcatSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4480: { ConcatLeftSep is same to previous function, but   strings concatenate right to left }
4481: function ConcatLeftSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4482:
4483: { Next 4 function for russian chars transliterating.
4484:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes suck }
4485: procedure Dos2Win(var S: AnsiString);
4486: procedure Win2Dos(var S: AnsiString);
4487: function Dos2WinRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4488: function Win2DosRes(const S: AnsiString): AnsiString; inline; {$ENDIF SUPPORTS_INLINE}
4489: function Win2Koi(const S: AnsiString): AnsiString;
4490: { FillString fills the string Buffer with Count Chars }
4491: procedure FillString(var Buffer: string; Count: Integer; const Value: Char); overload;
4492: procedure FillString(var Buffer: string; StartIndex, Count: Integer; const Value: Char); overload;
4493: { MoveString copies Count Chars from Source to Dest }
4494: procedure MoveString(const Source: string; var Dest: string; Count: Integer); {$IFDEF SUPPORTS_INLINE}
      inline; {$ENDIF SUPPORTS_INLINE} overload;
4495: procedure MoveString(const Source: string; SrcStartIdx: Integer; var Dest: string;
4496:   DstStartIdx: Integer;Count: Integer);{$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE} overload;
4497: { FillWideChar fills Buffer with Count WideChars (2 Bytes) }
4498: procedure FillWideChar(var Buffer; Count: Integer; const Value: WideChar);
4499: { MoveWideChar copies Count WideChars from Source to Dest }
4500: procedure MoveWideChar(const Source; var Dest;Count:Integer);{$IFDEF SUPPORTS_INLINE}inline;{$ENDIF
      SUPPORTS_INLINE}
4501: { FillNativeChar fills Buffer with Count NativeChars }
4502: procedure FillNativeChar(var Buffer; Count: Integer; const Value: Char); // D2009 internal error {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4503: { MoveWideChar copies Count WideChars from Source to Dest }
4504: procedure MoveNativeChar(const Source; var Dest; Count: Integer); // D2009 internal error {$IFDEF
      SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4505: { IsSubString() compares the sub string to the string. Indices are 1th based. }
4506: function IsSubString(const S: string; StartIndex: Integer; const SubStr: string): Boolean;
4507: { Spaces returns string consists on N space chars }
4508: function Spaces(const N: Integer): string;
4509: { AddSpaces adds spaces to string S, if its Length is smaller than N }
4510: function AddSpaces(const S: string; const N: Integer): string;
4511: function SpacesW(const N: Integer): WideString;
4512: function AddSpacesW(const S: WideString; const N: Integer): WideString;
4513: { function LastDateRUS for russian users only }
4514: { returns date relative to current date: 'ääà äíÿ íàçàä' }
4515: function LastDateRUS(const Dat: TDateTime): string;
4516: { CurrencyToStr format Currency, Cur, using ffCurrency float format}
4517: function CurrencyToStr(const Cur: Currency): string;
4518: { HasChar returns True, if Char, Ch, contains in string, S }
4519: function HasChar(const Ch: Char; const S: string): Boolean;
4520: function HasCharW(const Ch: WideChar; const S: WideString): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4521: function HasAnyChar(const Chars: string; const S: string): Boolean;
4522: {$IFNDEF COMPILER12_UP}
4523: function CharInSet(const Ch: AnsiChar;const SetOfChar:TSysCharSet):Boolean;inline;{$ENDIF SUPPORTS_INLINE}
4524: {$ENDIF ~COMPILER12_UP}
4525: function CharInSetW(const Ch: WideChar;const SetOfChar: TSysCharSet):Boolean;inline; {$ENDIF
      SUPPORTS_INLINE}
4526: function CountOfChar(const Ch: Char; const S: string): Integer;
4527: function DefStr(const S: string; Default: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
```

```
4528: { StrLICompW2 is a faster replacement for JclUnicode.StrLICompW }
4529: function StrLICompW2(S1, S2: PWideChar; MaxLen: Integer): Integer;
4530: function StrPosW(S, SubStr: PWideChar): PWideChar;
4531: function StrLenW(S: PWideChar): Integer;
4532: function TrimW(const S: WideString):WideString;{$IFDEF SUPPORTS_INLINE} inline;{$ENDIF SUPPORTS_INLINE}
4533: function TrimLeftW(const S: WideString): WideString; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
      SUPPORTS_INLINE}
4534: function TrimRightW(const S: WideString): WideString; inline; {$ENDIF SUPPORTS_INLINE}
4535: TPixelFormat', '( pfDevice, pf1bit, pf4bit, pf8bit, pf24bit )
4536: TMappingMethod', '( mmHistogram, mmQuantize, mmTrunc784, mmTrunc666, mmTripel, mmGrayscale )
4537:  Function GetBitmapPixelFormat( Bitmap : TBitmap) : TPixelFormat
4538: Function GetPaletteBitmapFormat( Bitmap : TBitmap) : TPixelFormat
4539: Procedure SetBitmapPixelFormat( Bitmap: TBitmap; PixelFormat: TPixelFormat; Method: TMappingMethod)
4540: Function BitmapToMemoryStream(Bitmap:TBitmap;PixelFormat:TPixelFormat;Method:TMappingMethod):TMemoryStream;
4541: Procedure GrayscaleBitmap( Bitmap : TBitmap)
4542: Function BitmapToMemory( Bitmap : TBitmap; Colors : Integer) : TStream
4543: Procedure SaveBitmapToFile( const Filename : string; Bitmap : TBitmap; Colors : Integer)
4544: Function ScreenPixelFormat : TPixelFormat
4545: Function ScreenColorCount : Integer
4546: Procedure TileImage( Canvas : TCanvas; Rect : TRect; Image : TGraphic)
4547: Function ZoomImage( ImageW, ImageH, MaxW, MaxH : Integer; Stretch : Boolean) : TPoint
4548: //  SIRegister_TJvGradient(CL);
4549:
4550: {************************************* files routines}
4551: procedure SetDelimitedText(List: TStrings; const Text: string; Delimiter: Char);
4552: const
4553:   {$IFDEF MSWINDOWS}
4554:   DefaultCaseSensitivity = False;
4555:   {$ENDIF MSWINDOWS}
4556:   {$IFDEF UNIX}
4557:   DefaultCaseSensitivity = True;
4558:   {$ENDIF UNIX}
4559: { GenTempFileName returns temporary file name on
4560:   drive, there FileName is placed }
4561: function GenTempFileName(FileName: string): string;
4562: { GenTempFileNameExt same to previous function, but
4563:   returning filename has given extension, FileExt }
4564: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
4565: { ClearDir clears folder Dir }
4566: function ClearDir(const Dir: string): Boolean;
4567: { DeleteDir clears and than delete folder Dir }
4568: function DeleteDir(const Dir: string): Boolean;
4569: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4570: function FileEquMask(FileName, Mask: TFileName; CaseSensitive: Boolean=DefaultCaseSensitivity): Boolean;
4571: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
4572:   Masks must be separated with SepPath (MSW: ';' / UNIX: ':') }
4573: function FileEquMasks(FileName, Masks: TFileName;
4574:   CaseSensitive: Boolean = DefaultCaseSensitivity): Boolean;
4575: function DeleteFiles(const Folder: TFileName; const Masks: string): Boolean;
4576: {$IFDEF MSWINDOWS}
4577: { LZFileExpand expand file, FileSource,
4578:   into FileDest. Given file must be compressed, using MS Compress program }
4579: function LZFileExpand(const FileSource, FileDest: string): Boolean;
4580: {$ENDIF MSWINDOWS}
4581: { FileGetInfo fills SearchRec record for specified file attributes}
4582: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
4583: { HasSubFolder returns True, if folder APath contains other folders }
4584: function HasSubFolder(APath: TFileName): Boolean;
4585: { IsEmptyFolder returns True, if there are no files or
4586:   folders in given folder, APath}
4587: function IsEmptyFolder(APath: TFileName): Boolean;
4588: { AddSlash returns string with added slash Char to Dir parameter, if needed }
4589: function AddSlash(const Dir: TFileName): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4590: { AddPath returns FileName with Path, if FileName not contain any path }
4591: function AddPath(const FileName, Path: TFileName): TFileName;
4592: function AddPaths(const PathList, Path: string): string;
4593: function ParentPath(const Path: TFileName): TFileName;
4594: function FindInPath(const FileName, PathList: string): TFileName;
4595: { DeleteReadOnlyFile clears R/O file attribute and delete file }
4596: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
4597: { HasParam returns True, if program running with specified parameter, Param }
4598: function HasParam(const Param: string): Boolean;
4599: function HasSwitch(const Param: string): Boolean;
4600: function Switch(const Param: string): string;
4601: { ExePath returns ExtractFilePath(ParamStr(0)) }
4602: function ExePath: TFileName; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4603: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
4604: //function FileTimeToDateTime(const FT: TFileTime): TDateTime;
4605: procedure FileTimeToDosDateTimeDWord(const FT: TFileTime; out Dft: DWORD);
4606: function MakeValidFileName(const FileName: TFileName; ReplaceBadChar: Char): TFileName;
4607: {**** Graphic routines }
4608: { IsTTFontSelected returns True, if True Type font is selected in specified device context }
4609: function IsTTFontSelected(const DC: HDC): Boolean;
4610: function KeyPressed(VK: Integer): Boolean;
4611: Function isKeypressed: boolean;  //true if key on memo2 (shell output) is pressed
4612: { TrueInflateRect inflates rect in other method, than InflateRect API function }
4613: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
4614: {**** Color routines }
4615: procedure RGBToHSV(R, G, B: Integer; var H, S, V: Integer);
```

```
4616: function RGBToBGR(Value: Cardinal): Cardinal;
4617: //function ColorToPrettyName(Value: TColor): string;
4618: //function PrettyNameToColor(const Value: string): TColor;
4619: {**** other routines }
4620: procedure SwapInt(var Int1, Int2: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4621: function IntPower(Base, Exponent: Integer): Integer;
4622: function ChangeTopException(E: TObject): TObject; // Linux version writes error message to ErrOutput
4623: function StrToBool(const S: string): Boolean;
4624: function Var2Type(V: Variant; const DestVarType: Integer): Variant;
4625: function VarToInt(V: Variant): Integer;
4626: function VarToFloat(V: Variant): Double;
4627: { following functions are not documented because they not work properly sometimes, so do not use them }
4628: // (rom) ReplaceStrings1, GetSubStr removed
4629: function GetLongFileName(const FileName: string): string;
4630: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
4631: function GetParameter: string;
4632: function GetComputerID: string;
4633: function GetComputerName: string;
4634: {**** string routines }
4635: { ReplaceAllStrings searches for all substrings, Words,
4636:   in a string, S, and replaces them with Frases with the same Index. }
4637: function ReplaceAllStrings(const S: string; Words, Frases: TStrings): string;
4638: { ReplaceStrings searches the Word in a string, S, on PosBeg position,
4639:   in the list, Words, and if founds, replaces this Word with string from another list,Frases, with the
     same Index, and then update NewSelStart variable }
4640: function ReplaceStrings(const S:string;PosBeg,Len:Int;Words,Frases:TStrings;var NewSelStart:Int):string;
4641: { CountOfLines calculates the lines count in a string, S,
4642:   each line must be separated from another with CrLf sequence }
4643: function CountOfLines(const S: string): Integer;
4644: { DeleteLines deletes all lines from strings which in the words,  words.
4645:   The word of will be deleted from strings. }
4646: procedure DeleteOfLines(Ss: TStrings; const Words: array of string);
4647: { DeleteEmptyLines deletes all empty lines from strings, Ss.
4648:   Lines contained only spaces also deletes. }
4649: procedure DeleteEmptyLines(Ss: TStrings);
4650: { SQLAddWhere addes or modifies existing where-statement, where,
4651:   to the strings, SQL. Note: If strings SQL allready contains where-statement,
4652:   it must be started on the begining of any line }
4653: procedure SQLAddWhere(SQL: TStrings; const Where: string);
4654: {**** files routines - }
4655: {$IFDEF MSWINDOWS}
4656: { ResSaveToFile save resource named as Name with Typ type into file FileName.
4657:   Resource can be compressed using MS Compress program}
4658: function ResSaveToFile(const Typ,Name:string; const Compressed: Boolean; const FileName: string): Boolean;
4659: function ResSaveToFileEx(Instance:HINST;Typ,Name:PChar;const Compressed:Boolean;const FileName:string):
     Boolean;
4660: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
4661: {$ENDIF MSWINDOWS}
4662: { IniReadSection read section, Section, from ini-file,
4663:   IniFileName, into strings, Ss.This function reads ALL strings from specified section.
4664:   Note: TIninFile.ReadSection function reads only strings with '=' symbol.}
4665: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
4666: { LoadTextFile load text file, FileName, into string }
4667: function LoadTextFile(const FileName: TFileName): string;
4668: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4669: { ReadFolder reads files list from disk folder,Folder,that are equal to mask, Mask,into strings, FileList}
4670: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
4671: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
4672: { RATextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
4673: procedure RATextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
4674: { RATextOutEx same with RATextOut function, but can calculate needed height for correct output }
4675: function RATextOutEx(Canvas:TCanvas;const R,RClip:TRect; const S:string;const CalcHeight:Boolean):Integer;
4676: { RATextCalcHeight calculate needed height for
4677:   correct output, using RATextOut or RATextOutEx functions }
4678: function RATextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
4679: { Cinema draws some visual effect }
4680: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
4681: { Roughed fills rect with special 3D pattern }
4682: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
4683: { 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 }
4684: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
4685: { TextWidth calculate text with for writing using standard desktop font }
4686: function TextWidth(const AStr: string): Integer;
4687: { TextHeight calculate text height for writing using standard desktop font }
4688: function TextHeight(const AStr: string): Integer;
4689: procedure SetChildPropOrd(Owner: TComponent; const PropName: string; Value: Longint);
4690: procedure Error(const Msg: string);
4691: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
4692:   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
4693: {example Text parameter:'Item 1<b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green<c:blue>blue</i>' }
4694: function ItemHtDraw(Canvas: TCanvas; Rect: TRect;
4695:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): string;
4696: function ItemHtWidth(Canvas: TCanvas; Rect: TRect;
4697:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): Integer;
4698: function ItemHtPlain(const Text: string): string;
4699: { ClearList - clears list of TObject }
4700: procedure ClearList(List: TList);
4701: procedure MemStreamToClipBoard(MemStream: TMemoryStream; const Format: Word);
```

```
4702: procedure ClipBoardToMemStream(MemStream: TMemoryStream; const Format: Word);
4703: { RTTI support }
4704: function GetPropType(Obj: TObject; const PropName: string): TTypeKind;
4705: function GetPropStr(Obj: TObject; const PropName: string): string;
4706: function GetPropOrd(Obj: TObject; const PropName: string): Integer;
4707: function GetPropMethod(Obj: TObject; const PropName: string): TMethod;
4708: procedure PrepareIniSection(Ss: TStrings);
4709: { following functions are not documented because they are don't work properly, so don't use them }
4710: // (rom) from JvBandWindows to make it obsolete
4711: function PointL(const X, Y: Longint): TPointL; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4712: // (rom) from JvBandUtils to make it obsolete
4713: function iif(const Test: Boolean; const ATrue, AFalse: Variant): Variant;
4714: procedure CopyIconToClipboard(Icon: TIcon; BackColor: TColor);
4715: function CreateIconFromClipboard: TIcon;
4716: { begin JvIconClipboardUtils } { Icon clipboard routines }
4717: function CF_ICON: Word;
4718: procedure AssignClipboardIcon(Icon: TIcon);
4719: { Real-size icons support routines (32-bit only) }
4720: procedure GetIconSize(Icon: HICON; var W, H: Integer);
4721: function CreateRealSizeIcon(Icon: TIcon): HICON;
4722: procedure DrawRealSizeIcon(Canvas: TCanvas; Icon: TIcon; X, Y: Integer);
4723: {end JvIconClipboardUtils }
4724: function CreateScreenCompatibleDC: HDC;
4725: function InvalidateRect(hWnd: HWND; const lpRect: TRect; bErase: BOOL): BOOL; overload; {$IFDEF
     SUPPORTS_INLINE} inline; {$ENDIF}
4726: function InvalidateRect(hWnd: HWND; lpRect: PRect; bErase: BOOL): BOOL; overload;
4727: { begin JvRLE } // (rom) changed API for inclusion in JCL
4728: procedure RleCompressTo(InStream, OutStream: TStream);
4729: procedure RleDecompressTo(InStream, OutStream: TStream);
4730: procedure RleCompress(Stream: TStream);
4731: procedure RleDecompress(Stream: TStream);
4732: { end JvRLE } { begin JvDateUtil }
4733: function CurrentYear: Word; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4734: function IsLeapYear(AYear: Integer): Boolean;
4735: function DaysInAMonth(const AYear, AMonth: Word): Word;
4736: function DaysPerMonth(AYear, AMonth: Integer): Integer;
4737: function FirstDayOfPrevMonth: TDateTime;
4738: function LastDayOfPrevMonth: TDateTime;
4739: function FirstDayOfNextMonth: TDateTime;
4740: function ExtractDay(ADate: TDateTime): Word;
4741: function ExtractMonth(ADate: TDateTime): Word;
4742: function ExtractYear(ADate: TDateTime): Word;
4743: function IncDate(ADate: TDateTime; Days, Months, Years: Integer): TDateTime;
4744: function IncDay(ADate: TDateTime;Delta:Integer):TDateTime; inline; {$ENDIF SUPPORTS_INLINE}
4745: function IncMonth(ADate: TDateTime; Delta: Integer): TDateTime;
4746: function IncYear(ADate: TDateTime; Delta: Integer): TDateTime;
4747: function ValidDate(ADate: TDateTime): Boolean;
4748: procedure DateDiff(Date1, Date2: TDateTime; var Days, Months, Years: Word);
4749: function MonthsBetween(Date1, Date2: TDateTime): Double;
4750: function DaysInPeriod(Date1, Date2: TDateTime): Longint;
4751: { Count days between Date1 and Date2 + 1, so if Date1 = Date2 result = 1 }
4752: function DaysBetween(Date1, Date2: TDateTime): Longint;
4753: { The same as previous but if Date2 < Date1 result = 0 }
4754: function IncTime(ATime: TDateTime; Hours, Minutes, Seconds, MSecs: Integer): TDateTime;
4755: function IncHour(ATime: TDateTime; Delta: Integer): TDateTime;
4756: function IncMinute(ATime: TDateTime; Delta: Integer): TDateTime;
4757: function IncSecond(ATime: TDateTime; Delta: Integer): TDateTime;
4758: function IncMSec(ATime: TDateTime; Delta: Integer): TDateTime;
4759: function CutTime(ADate: TDateTime): TDateTime;  { Set time to 00:00:00:00 }
4760: { String to date conversions }
4761: function GetDateOrder(const DateFormat: string): TDateOrder;
4762: function MonthFromName(const S: string; MaxLen: Byte): Byte;
4763: function StrToDateDef(const S: string; Default: TDateTime): TDateTime;
4764: function StrToDateFmt(const DateFormat, S: string): TDateTime;
4765: function StrToDateFmtDef(const DateFormat, S: string; Default: TDateTime): TDateTime;
4766: //function DefDateFormat(AFourDigitYear: Boolean): string;
4767: //function DefDateMask(BlanksChar: Char; AFourDigitYear: Boolean): string;
4768: function FormatLongDate(Value: TDateTime): string;
4769: function FormatLongDateTime(Value: TDateTime): string;
4770: { end JvDateUtil }
4771: function BufToBinStr(Buf: Pointer; BufSize: Integer): string;
4772: function BinStrToBuf(Value: string; Buf: Pointer; BufSize: Integer): Integer;
4773: { begin JvStrUtils } { ** Common string handling routines ** }
4774: {$IFDEF UNIX}
4775: function iconversion(InP: PAnsiChar; OutP: Pointer; InBytes, OutBytes: Cardinal;
4776:   const ToCode, FromCode: AnsiString): Boolean;
4777: function iconvString(const S, ToCode, FromCode: AnsiString): string;
4778: function iconvWideString(const S: WideString; const ToCode, FromCode: AnsiString): WideString;
4779: function OemStrToAnsi(const S: AnsiString): AnsiString;
4780: function AnsiStrToOem(const S: AnsiString): AnsiString;
4781: {$ENDIF UNIX}
4782: function StrToOem(const AnsiStr: AnsiString): AnsiString;
4783: { StrToOem translates a string from the Windows character set into the OEM character set. }
4784: function OemToAnsiStr(const OemStr: AnsiString): AnsiString;
4785: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
4786: function IsEmptyStr(const S: string; const EmptyChars: TSysCharSet): Boolean;
4787: { EmptyStr returns True if the given string contains only character from the EmptyChars. }
4788: function ReplaceStr(const S, Srch, Replace: string): string;
4789: { Returns string with every occurrence of Srch string replaced with Replace string. }
```

```
4790: function DelSpace(const S: string): string;
4791: { DelSpace return a string with all white spaces removed. }
4792: function DelChars(const S: string; Chr: Char): string;
4793: { DelChars return a string with all Chr characters removed. }
4794: function DelBSpace(const S: string): string;
4795: { DelBSpace trims leading spaces from the given string. }
4796: function DelESpace(const S: string): string;
4797: { DelESpace trims trailing spaces from the given string. }
4798: function DelRSpace(const S: string): string;
4799: { DelRSpace trims leading and trailing spaces from the given string. }
4800: function DelSpace1(const S: string): string;
4801: { DelSpace1 return a string with all non-single white spaces removed. }
4802: function Tab2Space(const S: string; Numb: Byte): string;
4803: { Tab2Space converts any tabulation character in the given string to the
4804:   Numb spaces characters. }
4805: function NPos(const C: string; S: string; N: Integer): Integer;
4806: { NPos searches for a N-th position of substring C in a given string. }
4807: function MakeStr(C: Char; N: Integer): string; overload;
4808: {$IFNDEF COMPILER12_UP}
4809: function MakeStr(C: WideChar; N: Integer): WideString; overload;
4810: {$ENDIF !COMPILER12_UP}
4811: function MS(C: Char; N: Integer): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4812: { MakeStr return a string of length N filled with character C. }
4813: function AddChar(C: Char; const S: string; N: Integer): string;
4814: { AddChar return a string left-padded to length N with characters C. }
4815: function AddCharR(C: Char; const S: string; N: Integer): string;
4816: { AddCharR return a string right-padded to length N with characters C. }
4817: function LeftStr(const S: string; N: Integer): string;
4818: { LeftStr return a string right-padded to length N with blanks. }
4819: function RightStr(const S: string; N: Integer): string;
4820: { RightStr return a string left-padded to length N with blanks. }
4821: function CenterStr(const S: string; Len: Integer): string;
4822: { CenterStr centers the characters in the string based upon the Len specified. }
4823: function CompStr(const S1, S2: string): Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4824: { CompStr compares S1 to S2, with case-sensitivity. The return value is
4825:   -1 if S1 < S2, 0 if S1 = S2, or 1 if S1 > S2. }
4826: function CompText(const S1, S2: string):Integer; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4827: { CompText compares S1 to S2, without case-sensitivity. The return value is the same as for CompStr. }
4828: function Copy2Symb(const S: string; Symb: Char): string;
4829: { Copy2Symb returns a substring of a string S from begining to first character Symb. }
4830: function Copy2SymbDel(var S: string; Symb: Char): string;
4831: { Copy2SymbDel returns a substring of a string S from begining to first
4832:   character Symb and removes this substring from S. }
4833: function Copy2Space(const S: string): string;
4834: { Copy2Symb returns a substring of a string S from begining to first white space. }
4835: function Copy2SpaceDel(var S: string): string;
4836: { Copy2SpaceDel returns a substring of a string S from begining to first
4837:   white space and removes this substring from S. }
4838: function AnsiProperCase(const S: string; const WordDelims: TSysCharSet): string;
4839: { Returns string, with the first letter of each word in uppercase,
4840:   all other letters in lowercase. Words are delimited by WordDelims. }
4841: function WordCount(const S: string; const WordDelims: TSysCharSet): Integer;
4842: { WordCount given a set of word delimiters, returns number of words in S. }
4843: function WordPosition(const N: Integer; const S: string; const WordDelims: TSysCharSet): Integer;
4844: { Given a set of word delimiters, returns start position of N'th word in S. }
4845: function ExtractWord(N: Integer; const S: string; const WordDelims: TSysCharSet): string;
4846: function ExtractWordPos(N: Integer;const S: string;const WordDelims:TSysCharSet;var Pos: Integer): string;
4847: function ExtractDelimited(N: Integer; const S: string;const Delims: TSysCharSet): string;
4848: { ExtractWord, ExtractWordPos and ExtractDelimited given a set of word
4849:   delimiters, return the N'th word in S. }
4850: function ExtractSubstr(const S: string; var Pos: Integer;const Delims: TSysCharSet): string;
4851: { ExtractSubstr given a set of word delimiters, returns the substring from S,
4852:   that started from position Pos. }
4853: function IsWordPresent(const W, S: string; const WordDelims: TSysCharSet): Boolean;
4854: { IsWordPresent given a set of word delimiters, returns True if word W is present in string S. }
4855: function QuotedString(const S: string; Quote: Char): string;
4856: { QuotedString returns the given string as a quoted string, using the provided Quote character. }
4857: function ExtractQuotedString(const S: string; Quote: Char): string;
4858: { ExtractQuotedString removes the Quote characters from the beginning and
4859:   end of a quoted string, and reduces pairs of Quote characters within quoted string to single character.}
4860: function FindPart(const HelpWilds, InputStr: string): Integer;
4861: { FindPart compares a string with '?' and another, returns the position of HelpWilds in InputStr. }
4862: function IsWild(InputStr, Wilds: string; IgnoreCase: Boolean): Boolean;
4863: { IsWild compares InputString with WildCard string and returns True if corresponds. }
4864: function XorString(const Key, Src: ShortString): ShortString;
4865: function XorEncode(const Key, Source: string): string;
4866: function XorDecode(const Key, Source: string): string;
4867: { ** Command line routines ** }
4868: function GetCmdLineArg(const Switch: string; ASwitchChars: TSysCharSet): string;
4869: { ** Numeric string handling routines ** }
4870: function Numb2USA(const S: string): string;
4871: { Numb2USA converts numeric string S to USA-format. }
4872: function Dec2Hex(N: Longint; A: Byte): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4873: { Dec2Hex converts the given value to a hexadecimal string representation
4874:   with the minimum number of digits (A) specified. }
4875: function Hex2Dec(const S: string): Longint;
4876: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4877: function Dec2Numb(N: Int64; A, B: Byte): string;
4878: { Dec2Numb converts the given value to a string representation with the
```

```
4879:   base equal to B and with the minimum number of digits (A) specified. }
4880: function Numb2Dec(S: string; B: Byte): Int64;
4881: { Numb2Dec converts the given B-based numeric string to the corresponding
4882:   integer value. }
4883: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4884: { IntToBin converts the given value to a binary string representation
4885:   with the minimum number of digits specified. }
4886: function IntToRoman(Value: Longint): string;
4887: { IntToRoman converts the given value to a roman numeric string representation. }
4888: function RomanToInt(const S: string): Longint;
4889: { RomanToInt converts the given string to an integer value. If the string
4890:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4891: function FindNotBlankCharPos(const S: string): Integer;
4892: function FindNotBlankCharPosW(const S: WideString): Integer;
4893: function AnsiChangeCase(const S: string): string;
4894: function WideChangeCase(const S: string): string;
4895: function StartsText(const SubStr, S: string): Boolean;
4896: function EndsText(const SubStr, S: string): Boolean;
4897: function DequotedStr(const S: string; QuoteChar: Char = ''''): string;
4898: function AnsiDequotedStr(const S: string; AQuote: Char): string; //follow Delphi 2009's "Ansi" prefix
4899: {end JvStrUtils}
4900: {$IFDEF UNIX}
4901: function GetTempFileName(const Prefix: AnsiString): AnsiString;
4902: {$ENDIF UNIX}
4903: { begin JvFileUtil }
4904: function FileDateTime(const FileName: string): TDateTime;
4905: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4906: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4907: function NormalDir(const DirName: string): string;
4908: function RemoveBackSlash(const DirName: string): string; // only for Windows/DOS Paths
4909: function ValidFileName(const FileName: string): Boolean;
4910: {$IFDEF MSWINDOWS}
4911: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4912: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4913: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4914: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4915: {$ENDIF MSWINDOWS}
4916: function GetWindowsDir: string;
4917: function GetSystemDir: string;
4918: function ShortToLongFileName(const ShortName: string): string;
4919: function LongToShortFileName(const LongName: string): string;
4920: function ShortToLongPath(const ShortName: string): string;
4921: function LongToShortPath(const LongName: string): string;
4922: {$IFDEF MSWINDOWS}
4923: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4924: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4925: {$ENDIF MSWINDOWS}
4926: { end JvFileUtil }
4927: // Works like PtInRect but includes all edges in comparision
4928: function PtInRectInclusive(R: TRect; Pt: TPoint): Boolean;
4929: // Works like PtInRect but excludes all edges from comparision
4930: function PtInRectExclusive(R: TRect; Pt: TPoint): Boolean;
4931: function FourDigitYear: Boolean; {$IFDEF SUPPORTS_DEPRECATED} deprecated; {$ENDIF}
4932: function IsFourDigitYear: Boolean;
4933: { moved from JvJVCLUTils }
4934: //Open an object with the shell (url or something like that)
4935: function OpenObject(const Value: string): Boolean; overload;
4936: function OpenObject(Value: PChar): Boolean; overload;
4937: {$IFDEF MSWINDOWS}
4938: //Raise the last Exception
4939: procedure RaiseLastWin32; overload;
4940: procedure RaiseLastWin32(const Text: string); overload;
4941: //Raise the last Exception with a small comment from your part { GetFileVersion returns the most
      significant 32 bits of a file's binary version number. Typically, this includes the major and minor
      version placed together in one 32-bit Integer. I
4942: function GetFileVersion(const AFileName: string): Cardinal;
4943: {$EXTERNALSYM GetFileVersion}
4944: //Get version of Shell.dll
4945: function GetShellVersion: Cardinal;
4946: {$EXTERNALSYM GetShellVersion}
4947: // CD functions  on HW
4948: procedure OpenCdDrive;
4949: procedure CloseCdDrive;
4950: // returns True if Drive is accessible
4951: function DiskInDrive(Drive: Char): Boolean;
4952: {$ENDIF MSWINDOWS}
4953: //Same as linux function ;)
4954: procedure PError(const Text: string);
4955: // execute a program without waiting
4956: procedure Exec(const FileName, Parameters, Directory: string);
4957: // execute a program and wait for it to finish
4958: function ExecuteAndWait(CmdLine:string;const WorkingDirectory:string;Visibility:Integer=SW_SHOW): Int;
4959: // returns True if this is the first instance of the program that is running
4960: function FirstInstance(const ATitle: string): Boolean;
4961: // restores a window based on it's classname and Caption. Either can be left empty
4962: // to widen the search
4963: procedure RestoreOtherInstance(const MainFormClassName, MainFormCaption: string);
4964: // manipulate the traybar and start button
4965: procedure HideTraybar;
```

```
4966: procedure ShowTraybar;
4967: procedure ShowStartButton(Visible: Boolean = True);
4968: // (rom) SC_MONITORPOWER is documented as Win 95 only(rom) better do some testing set monitor functions
4969: procedure MonitorOn;
4970: procedure MonitorOff;
4971: procedure LowPower;
4972: // send a key to the window named AppName
4973: function SendKey(const AppName: string; Key: Char): Boolean;
4974: {$IFDEF MSWINDOWS}
4975: // returns a list of all win currently visible, the Objects property is filled with their window handle
4976: procedure GetVisibleWindows(List: TStrings);
4977: // associates an extension to a specific program
4978: procedure AssociateExtension(const IconPath, ProgramName, Path, Extension: string);
4979: procedure AddToRecentDocs(const FileName: string);
4980: function GetRecentDocs: TStringList;
4981: {$ENDIF MSWINDOWS}
4982: function CharIsMoney(const Ch: Char): Boolean;
4983: //function StrToCurrDef(const Str: string; Def: Currency): Currency;
4984: function IntToExtended(I: Integer): Extended;
4985: { GetChangedText works out the new text given the current cursor pos & the key pressed
4986:   It is not very useful in other contexts, but in this unit as it is needed in both MemoEx and TypedEdit }
4987: function GetChangedText(const Text: string; SelStart, SelLength: Integer; Key: Char): string;
4988: function MakeYear4Digit(Year, Pivot: Integer): Integer;
4989: //function StrIsInteger(const S: string): Boolean;
4990: function StrIsFloatMoney(const Ps: string): Boolean;
4991: function StrIsDateTime(const Ps: string): Boolean;
4992: function PreformatDateString(Ps: string): string;
4993: function BooleanToInteger(const B: Boolean): Integer;
4994: function StringToBoolean(const Ps: string): Boolean;
4995: function SafeStrToDateTime(const Ps: string): TDateTime;
4996: function SafeStrToDate(const Ps: string): TDateTime;
4997: function SafeStrToTime(const Ps: string): TDateTime;
4998: function StrDelete(const psSub, psMain: string): string;
4999:   { returns the fractional value of pcValue}
5000: function TimeOnly(pcValue: TDateTime): TTime;
5001: { returns the integral value of pcValue }
5002: function DateOnly(pcValue: TDateTime): TDate;
5003: type TdtKind = (dtkDateOnly, dtkTimeOnly, dtkDateTime);
5004: const { TDateTime value used to signify Null value}
5005:   NullEquivalentDate: TDateTime = 0.0;
5006: function DateIsNull(const pdtValue: TDateTime; const pdtKind: TdtKind): Boolean;
5007: // Replacement for Win32Check to avoid platform specific warnings in D6
5008: function OSCheck(RetVal: Boolean): Boolean;
5009: { Shortens a fully qualified Path name so that it can be drawn with a specified length limit.
5010:   Same as FileCtrl.MinimizeName in functionality (but not implementation). Included here to
5011:   not be forced to use FileCtrl unnecessarily }
5012: function MinimizeFileName(const FileName: string; Canvas: TCanvas; MaxLen: Integer): string;
5013: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
5014: { MinimizeString trunactes long string, S, and appends'...'symbols,if Length of S is more than MaxLen }
5015: function MinimizeString(const S: string; const MaxLen: Integer): string;
5016: procedure RunDll32Internal(Wnd:THandle; const DLLName,FuncName,CmdLine:string;CmdShow:Integer=
     SW_SHOWDEFAULT);
5017: { GetDLLVersion loads DLLName, gets a pointer to DLLVersion function and calls it, returning major and
     minor version values from the function. Returns False if DLL not loaded or if GetDLLVersion couldn't be
     found.}
5018: function GetDLLVersion(const DLLName: string; var pdwMajor, pdwMinor: Integer): Boolean;
5019: {$ENDIF MSWINDOWS}
5020: procedure ResourceNotFound(ResID: PChar);
5021: function EmptyRect: TRect;
5022: function RectWidth(R: TRect): Integer;
5023: function RectHeight(R: TRect): Integer;
5024: function CompareRect(const R1, R2: TRect): Boolean;
5025: procedure RectNormalize(var R: TRect);
5026: function RectIsSquare(const R: TRect): Boolean;
5027: function RectSquare(var ARect: TRect; AMaxSize: Integer = -1): Boolean;
5028: //If AMaxSize = -1 ,then auto calc Square's max size
5029: {$IFDEF MSWINDOWS}
5030: procedure FreeUnusedOle;
5031: function GetWindowsVersion: string;
5032: function LoadDLL(const LibName: string): THandle;
5033: function RegisterServer(const ModuleName: string): Boolean;
5034: function UnregisterServer(const ModuleName: string): Boolean;
5035: {$ENDIF MSWINDOWS}
5036: { String routines }
5037: function GetEnvVar(const VarName: string): string;
5038: function AnsiUpperFirstChar(const S: string): string; //follow Delphi 2009's example with "Ansi" prefix
5039: function StringToPChar(var S: string): PChar;
5040: function StrPAlloc(const S: string): PChar;
5041: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
5042: function DropT(const S: string): string;
5043: { Memory routines }
5044: function AllocMemo(Size: Longint): Pointer;
5045: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
5046: procedure FreeMemo(var fpBlock: Pointer);
5047: function GetMemoSize(fpBlock: Pointer): Longint;
5048: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
5049: { Manipulate huge pointers routines }
5050: procedure HugeInc(var HugePtr: Pointer; Amount: Longint);
5051: procedure HugeDec(var HugePtr: Pointer; Amount: Longint);
```

```
5052: function HugeOffset(HugePtr: Pointer; Amount: Longint): Pointer;
5053: procedure HugeMove(Base: Pointer; Dst, Src, Size: Longint);
5054: procedure HMemCpy(DstPtr, SrcPtr: Pointer; Amount: Longint);
5055: function WindowClassName(Wnd: THandle): string;
5056: procedure SwitchToWindow(Wnd: THandle; Restore: Boolean);
5057: procedure ActivateWindow(Wnd: THandle);
5058: procedure ShowWinNoAnimate(Handle: THandle; CmdShow: Integer);
5059: procedure KillMessage(Wnd: THandle; Msg: Cardinal);
5060: { SetWindowTop put window to top without recreating window }
5061: procedure SetWindowTop(const Handle: THandle; const Top: Boolean);
5062: procedure CenterWindow(Wnd: THandle);
5063: function MakeVariant(const Values: array of Variant): Variant;
5064: { Convert dialog units to pixels and backwards }
5065: {$IFDEF MSWINDOWS}
5066: function DialogUnitsToPixelsX(DlgUnits: Word): Word;
5067: function DialogUnitsToPixelsY(DlgUnits: Word): Word;
5068: function PixelsToDialogUnitsX(PixUnits: Word): Word;
5069: function PixelsToDialogUnitsY(PixUnits: Word): Word;
5070: {$ENDIF MSWINDOWS}
5071: function GetUniqueFileNameInDir(const Path, FileNameMask: string): string;
5072: {$IFDEF BCB}
5073: function FindPrevInstance(const MainFormClass: ShortString;const ATitle: string): THandle;
5074: function ActivatePrevInstance(const MainFormClass: ShortString;const ATitle: string): Boolean;
5075: {$ELSE}
5076: function FindPrevInstance(const MainFormClass, ATitle: string): THandle;
5077: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
5078: {$ENDIF BCB}
5079: {$IFDEF MSWINDOWS}
5080: { BrowseForFolderNative displays Browse For Folder dialog }
5081: function BrowseForFolderNative(const Handle: THandle; const Title: string; var Folder: string): Boolean;
5082: {$ENDIF MSWINDOWS}
5083: procedure AntiAlias(Clip: TBitmap);
5084: procedure AntiAliasRect(Clip: TBitmap; XOrigin, YOrigin,XFinal, YFinal: Integer);
5085: procedure CopyRectDIBits(ACanvas: TCanvas; const DestRect: TRect;
5086:   ABitmap: TBitmap; const SourceRect: TRect);
5087: function IsTrueType(const FontName: string): Boolean;
5088: // Removes all non-numeric characters from AValue and returns the resulting string
5089: function TextToValText(const AValue: string): string;
5090: Function ExecRegExpr( const ARegExpr, AInputStr : RegExprString) : boolean
5091: Procedure SplitRegExpr( const ARegExpr, AInputStr : RegExprString; APieces : TStrings)
5092: Function ReplaceRegExpr(const ARegExpr,AInputStr,
     AReplaceStr:RegExprString;AUseSubstitution:bool):RegExprString;
5093: Function QuoteRegExprMetaChars( const AStr : RegExprString) : RegExprString
5094: Function RegExprSubExpressions(const ARegExpr:string; ASubExprs:TStrings; AExtendedSyntax : boolean) :
5095:
5096: ****************************************************unit uPSI_JvTFUtils;
5097:  Function JExtractYear( ADate : TDateTime) : Word
5098:  Function JExtractMonth( ADate : TDateTime) : Word
5099:  Function JExtractDay( ADate : TDateTime) : Word
5100:  Function ExtractHours( ATime : TDateTime) : Word
5101:  Function ExtractMins( ATime : TDateTime) : Word
5102:  Function ExtractSecs( ATime : TDateTime) : Word
5103:  Function ExtractMSecs( ATime : TDateTime) : Word
5104:  Function FirstOfMonth( ADate : TDateTime) : TDateTime
5105:  Function GetDayOfNthDOW( Year, Month, DOW, N : Word) : Word
5106:  Function GetWeeksInMonth( Year, Month : Word; StartOfWeek : Integer) : Word
5107:  Procedure IncBorlDOW( var BorlDOW : Integer; N : Integer)
5108:  Procedure IncDOW( var DOW : TTFDayOfWeek; N : Integer)
5109:  Procedure IncDays( var ADate : TDateTime; N : Integer)
5110:  Procedure IncWeeks( var ADate : TDateTime; N : Integer)
5111:  Procedure IncMonths( var ADate : TDateTime; N : Integer)
5112:  Procedure IncYears( var ADate : TDateTime; N : Integer)
5113:  Function EndOfMonth( ADate : TDateTime) : TDateTime
5114:  Function IsFirstOfMonth( ADate : TDateTime) : Boolean
5115:  Function IsEndOfMonth( ADate : TDateTime) : Boolean
5116:  Procedure EnsureMonth( Month : Word)
5117:  Procedure EnsureDOW( DOW : Word)
5118:  Function EqualDates( D1, D2 : TDateTime) : Boolean
5119:  Function Lesser( N1, N2 : Integer) : Integer
5120:  Function Greater( N1, N2 : Integer) : Integer
5121:  Function GetDivLength( TotalLength, DivCount, DivNum : Integer) : Integer
5122:  Function GetDivNum( TotalLength, DivCount, X : Integer) : Integer
5123:  Function GetDivStart( TotalLength, DivCount, DivNum : Integer) : Integer
5124:  Function DOWToBorl( ADOW : TTFDayOfWeek) : Integer
5125:  Function BorlToDOW( BorlDOW : Integer) : TTFDayOfWeek
5126:  Function DateToDOW( ADate : TDateTime) : TTFDayOfWeek
5127:  Procedure CalcTextPos( HostRect : TRect; var TextLeft, TextTop:Integer; var TextBounds : TRect;
     AFont:TFont;AAngle: Integer; HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5128:  Procedure DrawAngleText( ACanvas : TCanvas; HostRect : TRect; var TextBounds : TRect; AAngle : Integer;
     HAlign : TAlignment; VAlign : TJvTFVAlignment; ATxt : string)
5129:  Function JRectWidth( ARect : TRect) : Integer
5130:  Function JRectHeight( ARect : TRect) : Integer
5131:  Function JEmptyRect : TRect
5132:  Function IsClassByName( Obj : TObject; ClassName : string) : Boolean
5133:
5134: procedure SIRegister_MSysUtils(CL: TPSPascalCompiler);
5135: begin
5136:  Procedure HideTaskBarButton( hWindow : HWND)
5137:  Function msLoadStr( ID : Integer) : String
```

```
5138:  Function msFormat( fmt : String; params : array of const) : String
5139:  Function msFileExists( const FileName : String) : Boolean
5140:  Function msIntToStr( Int : Int64) : String
5141:  Function msStrPas( const Str : PChar) : String
5142:  Function msRenameFile( const OldName, NewName : String) : Boolean
5143:  Function CutFileName( s : String) : String
5144:  Function GetVersionInfo( var VersionString : String) : DWORD
5145:  Function FormatTime( t : Cardinal) : String
5146:  Function msCreateDir( const Dir : string) : Boolean
5147:  Function SetAutoRun( NeedAutoRun : Boolean; AppName : String) : Boolean
5148:  Function SetTreeViewStyle( const hTV : HWND; dwNewStyle : dword) : DWORD
5149:  Function msStrLen( Str : PChar) : Integer
5150:  Function msDirectoryExists( const Directory : String) : Boolean
5151:  Function GetFolder( hWnd : hWnd; RootDir : Integer; Caption : String) : String
5152:  Function SetBlendWindow( hWnd : HWND; AlphaBlend : Byte) : LongBool
5153:  Function EditWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
5154:  Procedure SetEditWndProc( hWnd : HWND; ptr : TObject)
5155:  Function GetTextFromFile( Filename : String) : string
5156:  Function IsTopMost( hWnd : HWND) : Bool // 'LWA_ALPHA','LongWord').SetUInt( $00000002);
5157:  Function msStrToIntDef( const s : String; const i : Integer) : Integer
5158:  Function msStrToInt( s : String) : Integer
5159:  Function GetItemText( hDlg : THandle; ID : DWORD) : String
5160: end;
5161:
5162: procedure SIRegister_ESBMaths2(CL: TPSPascalCompiler);
5163: begin
5164:   //TDynFloatArray', 'array of Extended
5165:   TDynLWordArray', 'array of LongWord
5166:   TDynLIntArray', 'array of LongInt
5167:   TDynFloatMatrix', 'array of TDynFloatArray
5168:   TDynLWordMatrix', 'array of TDynLWordArray
5169:   TDynLIntMatrix', 'array of TDynLIntArray
5170:  Function SquareAll( const X : TDynFloatArray) : TDynFloatArray
5171:  Function InverseAll( const X : TDynFloatArray) : TDynFloatArray
5172:  Function LnAll( const X : TDynFloatArray) : TDynFloatArray
5173:  Function Log10All( const X : TDynFloatArray) : TDynFloatArray
5174:  Function LinearTransform( const X : TDynFloatArray; Offset, Scale : Extended) : TDynFloatArray
5175:  Function AddVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5176:  Function SubVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5177:  Function MultVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5178:  Function DotProduct( const X, Y : TDynFloatArray) : Extended
5179:  Function MNorm( const X : TDynFloatArray) : Extended
5180:  Function MatrixIsRectangular( const X : TDynFloatMatrix) : Boolean
5181:  Procedure MatrixDimensions(const X:TDynFloatMatrix;var Rows,Columns:LongWord;var Rectangular:Boolean;
5182:  Function MatrixIsSquare( const X : TDynFloatMatrix) : Boolean
5183:  Function MatricesSameDimensions( const X, Y : TDynFloatMatrix) : Boolean
5184:  Function AddMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5185:  Procedure AddToMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5186:  Function SubtractMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5187:  Procedure SubtractFromMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5188:  Function MultiplyMatrixByConst( const X : TDynFloatMatrix; const K : Extended) : TDynFloatMatrix
5189:  Procedure MultiplyMatrixByConst2( var X : TDynFloatMatrix; const K : Extended);
5190:  Function MultiplyMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix;
5191:  Function TransposeMatrix( const X : TDynFloatMatrix) : TDynFloatMatrix;
5192:  Function GrandMean( const X : TDynFloatMatrix; var N : LongWord) : Extended
5193: end;
5194:
5195: procedure SIRegister_ESBMaths(CL: TPSPascalCompiler);
5196: begin
5197:  'ESBMinSingle','Single').setExtended( 1.5e-45);
5198:  'ESBMaxSingle','Single').setExtended( 3.4e+38);
5199:  'ESBMinDouble','Double').setExtended( 5.0e-324);
5200:  'ESBMaxDouble','Double').setExtended( 1.7e+308);
5201:  'ESBMinExtended','Extended').setExtended( 3.6e-4951);
5202:  'ESBMaxExtended','Extended').setExtended( 1.1e+4932);
5203:  'ESBMinCurrency','Currency').SetExtended( - 922337203685477.5807);
5204:  'ESBMaxCurrency','Currency').SetExtended( 922337203685477.5807);
5205:  'ESBSqrt2','Extended').setExtended( 1.4142135623730950488);
5206:  'ESBSqrt3','Extended').setExtended( 1.7320508075688772935);
5207:  'ESBSqrt5','Extended').setExtended( 2.2360679774997896964);
5208:  'ESBSqrt10','Extended').setExtended( 3.1622776601683793320);
5209:  'ESBSqrtPi','Extended').setExtended( 1.7724538509055160272 9);
5210:  'ESBCbrt2','Extended').setExtended( 1.2599210498948731648);
5211:  'ESBCbrt3','Extended').setExtended( 1.4422495703074083823);
5212:  'ESBCbrt10','Extended').setExtended( 2.1544346900318837219);
5213:  'ESBCbrt100','Extended').setExtended( 4.6415888336127788924);
5214:  'ESBCbrtPi','Extended').setExtended( 1.4645918875615232630);
5215:  'ESBInvSqrt2','Extended').setExtended( 0.70710678118654752440);
5216:  'ESBInvSqrt3','Extended').setExtended( 0.57735026918962576451);
5217:  'ESBInvSqrt5','Extended').setExtended( 0.44721359549995793928);
5218:  'ESBInvSqrtPi','Extended').setExtended( 0.56418958354775628695);
5219:  'ESBInvCbrtPi','Extended').setExtended( 0.68278406325529568147);
5220:  'ESBe','Extended').setExtended( 2.7182818284590452354);
5221:  'ESBe2','Extended').setExtended( 7.3890560989306502272);
5222:  'ESBePi','Extended').setExtended( 23.140692632779269006);
5223:  'ESBePiOn2','Extended').setExtended( 4.8104773809653516555);
5224:  'ESBePiOn4','Extended').setExtended( 2.1932800507380154566);
5225:  'ESBLn2','Extended').setExtended( 0.69314718055994530942);
5226:  'ESBLn10','Extended').setExtended( 2.3025850929940456840 2);
```

```
5227:  'ESBLnPi','Extended').setExtended( 1.14472988584940017414);
5228:  'ESBLog10Base2','Extended').setExtended( 3.3219280948873623478);
5229:  'ESBLog2Base10','Extended').setExtended( 0.30102999566398119521);
5230:  'ESBLog3Base10','Extended').setExtended( 0.47712125471966243730);
5231:  'ESBLogPiBase10','Extended').setExtended( 0.4971498726941339);
5232:  'ESBLogEBase10','Extended').setExtended( 0.43429448190325182765);
5233:  'ESBPi','Extended').setExtended( 3.1415926535897932385);
5234:  'ESBInvPi','Extended').setExtended( 3.1830988618379067154e-1);
5235:  'ESBTwoPi','Extended').setExtended( 6.2831853071795864769);
5236:  'ESBThreePi','Extended').setExtended( 9.4247779607693797153);
5237:  'ESBPi2','Extended').setExtended( 9.8696044010893586188);
5238:  'ESBPiToE','Extended').setExtended( 22.459157718361045473);
5239:  'ESBPiOn2','Extended').setExtended( 1.5707963267948966192);
5240:  'ESBPiOn3','Extended').setExtended( 1.0471975511965977462);
5241:  'ESBPiOn4','Extended').setExtended( 0.7853981633974483096);
5242:  'ESBThreePiOn2','Extended').setExtended( 4.7123889803846898577);
5243:  'ESBFourPiOn3','Extended').setExtended( 4.1887902047863909846);
5244:  'ESBTwoToPower63','Extended').setExtended( 9223372036854775808.0);
5245:  'ESBOneRadian','Extended').setExtended( 57.295779513082320877);
5246:  'ESBOneDegree','Extended').setExtended( 1.7453292519943295769E-2);
5247:  'ESBOneMinute','Extended').setExtended( 2.9088820866572159615E-4);
5248:  'ESBOneSecond','Extended').setExtended( 4.8481368110953599359E-6);
5249:  'ESBGamma','Extended').setExtended( 0.57721566490153286061);
5250:  'ESBLnRt2Pi','Extended').setExtended( 9.189385332046727E-1);
5251:   //LongWord', 'Cardinal
5252:   TBitList', 'Word
5253:  Function UMul( const Num1, Num2 : LongWord) : LongWord
5254:  Function UMulDiv2p32( const Num1, Num2 : LongWord) : LongWord
5255:  Function UMulDiv( const Num1, Num2, Divisor : LongWord) : LongWord
5256:  Function UMulMod( const Num1, Num2, Modulus : LongWord) : LongWord
5257:  Function SameFloat( const X1, X2 : Extended) : Boolean
5258:  Function FloatIsZero( const X : Extended) : Boolean
5259:  Function FloatIsPositive( const X : Extended) : Boolean
5260:  Function FloatIsNegative( const X : Extended) : Boolean
5261:  Procedure IncLim( var B : Byte; const Limit : Byte)
5262:  Procedure IncLimSI( var B : ShortInt; const Limit : ShortInt)
5263:  Procedure IncLimW( var B : Word; const Limit : Word)
5264:  Procedure IncLimI( var B : Integer; const Limit : Integer)
5265:  Procedure IncLimL( var B : LongInt; const Limit : LongInt)
5266:  Procedure DecLim( var B : Byte; const Limit : Byte)
5267:  Procedure DecLimSI( var B : ShortInt; const Limit : ShortInt)
5268:  Procedure DecLimW( var B : Word; const Limit : Word)
5269:  Procedure DecLimI( var B : Integer; const Limit : Integer)
5270:  Procedure DecLimL( var B : LongInt; const Limit : LongInt)
5271:  Function MaxB( const B1, B2 : Byte) : Byte
5272:  Function MinB( const B1, B2 : Byte) : Byte
5273:  Function MaxSI( const B1, B2 : ShortInt) : ShortInt
5274:  Function MinSI( const B1, B2 : ShortInt) : ShortInt
5275:  Function MaxW( const B1, B2 : Word) : Word
5276:  Function MinW( const B1, B2 : Word) : Word
5277:  Function esbMaxI( const B1, B2 : Integer) : Integer
5278:  Function esbMinI( const B1, B2 : Integer) : Integer
5279:  Function MaxL( const B1, B2 : LongInt) : LongInt
5280:  Function MinL( const B1, B2 : LongInt) : LongInt
5281:  Procedure SwapB( var B1, B2 : Byte)
5282:  Procedure SwapSI( var B1, B2 : ShortInt)
5283:  Procedure SwapW( var B1, B2 : Word)
5284:  Procedure SwapI( var B1, B2 : SmallInt)
5285:  Procedure SwapL( var B1, B2 : LongInt)
5286:  Procedure SwapI32( var B1, B2 : Integer)
5287:  Procedure SwapC( var B1, B2 : LongWord)
5288:  Procedure SwapInt64( var X, Y : Int64)
5289:  Function esbSign( const B : LongInt) : ShortInt
5290:  Function Max4Word( const X1, X2, X3, X4 : Word) : Word
5291:  Function Min4Word( const X1, X2, X3, X4 : Word) : Word
5292:  Function Max3Word( const X1, X2, X3 : Word) : Word
5293:  Function Min3Word( const X1, X2, X3 : Word) : Word
5294:  Function MaxBArray( const B : array of Byte) : Byte
5295:  Function MaxWArray( const B : array of Word) : Word
5296:  Function MaxSIArray( const B : array of ShortInt) : ShortInt
5297:  Function MaxIArray( const B : array of Integer) : Integer
5298:  Function MaxLArray( const B : array of LongInt) : LongInt
5299:  Function MinBArray( const B : array of Byte) : Byte
5300:  Function MinWArray( const B : array of Word) : Word
5301:  Function MinSIArray( const B : array of ShortInt) : ShortInt
5302:  Function MinIArray( const B : array of Integer) : Integer
5303:  Function MinLArray( const B : array of LongInt) : LongInt
5304:  Function SumBArray( const B : array of Byte) : Byte
5305:  Function SumBArray2( const B : array of Byte) : Word
5306:  Function SumSIArray( const B : array of ShortInt) : ShortInt
5307:  Function SumSIArray2( const B : array of ShortInt) : Integer
5308:  Function SumWArray( const B : array of Word) : Word
5309:  Function SumWArray2( const B : array of Word) : LongInt
5310:  Function SumIArray( const B : array of Integer) : Integer
5311:  Function SumLArray( const B : array of LongInt) : LongInt
5312:  Function SumLWArray( const B : array of LongWord) : LongWord
5313:  Function ESBDigits( const X : LongWord) : Byte
5314:  Function BitsHighest( const X : LongWord) : Integer
5315:  Function ESBBitsNeeded( const X : LongWord) : Integer
```

```
5316:  Function esbGCD( const X, Y : LongWord) : LongWord
5317:  Function esbLCM( const X, Y : LongInt) : Int64
5318:  //Function esbLCM( const X, Y : LongInt) : LongInt
5319:  Function RelativePrime( const X, Y : LongWord) : Boolean
5320:  Function Get87ControlWord : TBitList
5321:  Procedure Set87ControlWord( const CWord : TBitList)
5322:  Procedure SwapExt( var X, Y : Extended)
5323:  Procedure SwapDbl( var X, Y : Double)
5324:  Procedure SwapSing( var X, Y : Single)
5325:  Function esbSgn( const X : Extended) : ShortInt
5326:  Function Distance( const X1, Y1, X2, Y2 : Extended) : Extended
5327:  Function ExtMod( const X, Y : Extended) : Extended
5328:  Function ExtRem( const X, Y : Extended) : Extended
5329:  Function CompMOD( const X, Y : Comp) : Comp
5330:  Procedure Polar2XY( const Rho, Theta : Extended; var X, Y : Extended)
5331:  Procedure XY2Polar( const X, Y : Extended; var Rho, Theta : Extended)
5332:  Function DMS2Extended( const Degs, Mins, Secs : Extended) : Extended
5333:  Procedure Extended2DMS( const X : Extended; var Degs, Mins, Secs : Extended)
5334:  Function MaxExt( const X, Y : Extended) : Extended
5335:  Function MinExt( const X, Y : Extended) : Extended
5336:  Function MaxEArray( const B : array of Extended) : Extended
5337:  Function MinEArray( const B : array of Extended) : Extended
5338:  Function MaxSArray( const B : array of Single) : Single
5339:  Function MinSArray( const B : array of Single) : Single
5340:  Function MaxCArray( const B : array of Comp) : Comp
5341:  Function MinCArray( const B : array of Comp) : Comp
5342:  Function SumSArray( const B : array of Single) : Single
5343:  Function SumEArray( const B : array of Extended) : Extended
5344:  Function SumSqEArray( const B : array of Extended) : Extended
5345:  Function SumSqDiffEArray( const B : array of Extended; Diff : Extended) : Extended
5346:  Function SumXYEArray( const X, Y : array of Extended) : Extended
5347:  Function SumCArray( const B : array of Comp) : Comp
5348:  Function FactorialX( A : LongWord) : Extended
5349:  Function PermutationX( N, R : LongWord) : Extended
5350:  Function esbBinomialCoeff( N, R : LongWord) : Extended
5351:  Function IsPositiveEArray( const X : array of Extended) : Boolean
5352:  Function esbGeometricMean( const X : array of Extended) : Extended
5353:  Function esbHarmonicMean( const X : array of Extended) : Extended
5354:  Function ESBMean( const X : array of Extended) : Extended
5355:  Function esbSampleVariance( const X : array of Extended) : Extended
5356:  Function esbPopulationVariance( const X : array of Extended) : Extended
5357:  Procedure esbSampleVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5358:  Procedure esbPopulationVarianceAndMean( const X : array of Extended; var Variance, Mean : Extended)
5359:  Function GetMedian( const SortedX : array of Extended) : Extended
5360:  Function GetMode( const SortedX : array of Extended; var Mode : Extended) : Boolean
5361:  Procedure GetQuartiles( const SortedX : array of Extended; var Q1, Q3 : Extended)
5362:  Function ESBMagnitude( const X : Extended) : Integer
5363:  Function ESBTan( Angle : Extended) : Extended
5364:  Function ESBCot( Angle : Extended) : Extended
5365:  Function ESBCosec( const Angle : Extended) : Extended
5366:  Function ESBSec( const Angle : Extended) : Extended
5367:  Function ESBArcTan( X, Y : Extended) : Extended
5368:  Procedure ESBSinCos( Angle : Extended; var SinX, CosX : Extended)
5369:  Function ESBArcCos( const X : Extended) : Extended
5370:  Function ESBArcSin( const X : Extended) : Extended
5371:  Function ESBArcSec( const X : Extended) : Extended
5372:  Function ESBArcCosec( const X : Extended) : Extended
5373:  Function ESBLog10( const X : Extended) : Extended
5374:  Function ESBLog2( const X : Extended) : Extended
5375:  Function ESBLogBase( const X, Base : Extended) : Extended
5376:  Function Pow2( const X : Extended) : Extended
5377:  Function IntPow( const Base : Extended; const Exponent : LongWord) : Extended
5378:  Function ESBIntPower( const X : Extended; const N : LongInt) : Extended
5379:  Function XtoY( const X, Y : Extended) : Extended
5380:  Function esbTenToY( const Y : Extended) : Extended
5381:  Function esbTwoToY( const Y : Extended) : Extended
5382:  Function LogXtoBaseY( const X, Y : Extended) : Extended
5383:  Function esbISqrt( const I : LongWord) : Longword
5384:  Function ILog2( const I : LongWord) : LongWord
5385:  Function IGreatestPowerOf2( const N : LongWord) : LongWord
5386:  Function ESBArCosh( X : Extended) : Extended
5387:  Function ESBArSinh( X : Extended) : Extended
5388:  Function ESBArTanh( X : Extended) : Extended
5389:  Function ESBCosh( X : Extended) : Extended
5390:  Function ESBSinh( X : Extended) : Extended
5391:  Function ESBTanh( X : Extended) : Extended
5392:  Function InverseGamma( const X : Extended) : Extended
5393:  Function esbGamma( const X : Extended) : Extended
5394:  Function esbLnGamma( const X : Extended) : Extended
5395:  Function esbBeta( const X, Y : Extended) : Extended
5396:  Function IncompleteBeta( X : Extended; P, Q : Extended) : Extended
5397: end;
5398:
5399: ******************************Integer Huge Cardinal Utils*************************
5400: Function Add_uint64_WithCarry( x, y : uint64; var Carry : Boolean) : uint64
5401: Function Add_uint32_WithCarry( x, y : uint32; var Carry : Boolean) : uint32
5402: Function Subtract_uint64_WithBorrow( x, y : uint64; var Borrow : Boolean) : uint64
5403: Function Subtract_uint32_WithBorrow( x, y : uint32; var Borrow : Boolean) : uint32
5404: Function BitCount_8( Value : byte) : integer
```

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

```
5488:  Function XRTLToBin( const AInteger : TXRTLInteger; Digits : Integer) : string
5489:  Function XRTLToString( const AInteger : TXRTLInteger; Radix : Integer; Digits : Integer) : string
5490:  Procedure XRTLFromHex( const Value : string; var AResult : TXRTLInteger)
5491:  Procedure XRTLFromBin( const Value : string; var AResult : TXRTLInteger)
5492:  Procedure XRTLFromString( const Value : string; var AResult : TXRTLInteger; Radix : Integer)
5493:  Procedure XRTLAssign( const AInteger : TXRTLInteger; var AResult : TXRTLInteger);
5494:  Procedure XRTLAssign1( const Value : Integer; var AResult : TXRTLInteger);
5495:  Procedure XRTLAssign2( const Value : Int64; var AResult : TXRTLInteger);
5496:  Procedure XRTLAppend( const ALow, AHigh : TXRTLInteger; var AResult : TXRTLInteger)
5497:  Procedure XRTLSplit(const AInteger: TXRTLInteger; var ALow,AHigh: TXRTLInteger;LowDigits: Integer)
5498:  Function XRTLGetMSBitIndex( const AInteger : TXRTLInteger) : Integer
5499:  Procedure XRTLMinMax(const AInteger1, AInteger2 : TXRTLInteger;var AMinResult,AMaxResult: TXRTLInteger)
5500:  Procedure XRTLMin( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5501:  Procedure XRTLMin1(const AInteger1: TXRTLInteger;const AInteger2:Integer;var AResult : TXRTLInteger);
5502:  Procedure XRTLMax( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger);
5503:  Procedure XRTLMax1(const AInteger1:TXRTLInteger; const AInteger2:Integer; var AResult:TXRTLInteger);
5504:  Procedure XRTLGCD( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5505:  Procedure XRTLSwap( var AInteger1, AInteger2 : TXRTLInteger)
5506:  Procedure XRTLFactorial( const AInteger : TXRTLInteger; var AResult : TXRTLInteger)
5507:  Procedure XRTLFactorialMod( const AInteger1, AInteger2 : TXRTLInteger; var AResult : TXRTLInteger)
5508: end;
5509:
5510: procedure SIRegister_JvXPCoreUtils(CL: TPSPascalCompiler);
5511: begin
5512:  Function JvXPMethodsEqual( const Method1, Method2 : TMethod) : Boolean
5513:  Procedure JvXPDrawLine( const ACanvas : TCanvas; const X1, Y1, X2, Y2 : Integer)
5514:  Procedure JvXPCreateGradientRect( const AWidth, AHeight : Integer; const StartColor, EndColor : TColor;
      const Colors:TJvXPGradientColors;const Style:TJvXPGradientStyle;const Dithered:Boolean;var Bitmap:TBitmap);
5515:  Procedure JvXPAdjustBoundRect(const BorderWidth:Byte; const ShowBoundLines:Boolean; const
      BoundLines:TJvXPBoundLines; var Rect : TRect)
5516:  Procedure JvXPDrawBoundLines(const ACan:TCanvas;const BoundLines:TJvXPBoundLines;const
      AColor:TColor;const Rect:TRect);
5517:  Procedure JvXPConvertToGray2( Bitmap : TBitmap)
5518:  Procedure JvXPRenderText( const AParent : TControl; const ACanvas : TCanvas; ACaption : TCaption; const
      AFont : TFont; const AEnabled, AShowAccelChar : Boolean; var ARect : TRect; AFlags : Integer)
5519:  Procedure JvXPFrame3D(const ACanvas:TCanvas;const Rect:TRect;const TopColor,BottomColor:TColor;const
      Swapped:Boolean);
5520:  Procedure JvXPColorizeBitmap( Bitmap : TBitmap; const AColor : TColor)
5521:  Procedure JvXPSetDrawFlags(const AAlignment: TAlignment;const AWordWrap: Boolean; var Flags : Integer)
5522:  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)
5523: end;
5524:
5525:
5526: procedure SIRegister_uwinstr(CL: TPSPascalCompiler);
5527: begin
5528:  Function StrDec( S : String) : String
5529:  Function uIsNumeric( var S : String; var X : Float) : Boolean
5530:  Function ReadNumFromEdit( Edit : TEdit) : Float
5531:  Procedure WriteNumToFile( var F : Text; X : Float)
5532: end;
5533:
5534: procedure SIRegister_utexplot(CL: TPSPascalCompiler);
5535: begin
5536:  Function TeX_InitGraphics( FileName : String; PgWidth, PgHeight : Integer; Header : Boolean) : Boolean
5537:  Procedure TeX_SetWindow( X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
5538:  Procedure TeX_LeaveGraphics( Footer : Boolean)
5539:  Procedure TeX_SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
5540:  Procedure TeX_SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
5541:  Procedure TeX_SetGraphTitle( Title : String)
5542:  Procedure TeX_SetOxTitle( Title : String)
5543:  Procedure TeX_SetOyTitle( Title : String)
5544:  Procedure TeX_PlotOxAxis
5545:  Procedure TeX_PlotOyAxis
5546:  Procedure TeX_PlotGrid( Grid : TGrid)
5547:  Procedure TeX_WriteGraphTitle
5548:  Function TeX_SetMaxCurv( NCurv : Byte) : Boolean
5549:  Procedure TeX_SetPointParam( CurvIndex, Symbol, Size : Integer)
5550:  Procedure TeX_SetLineParam( CurvIndex, Style : Integer; Width : Float; Smooth : Boolean)
5551:  Procedure TeX_SetCurvLegend( CurvIndex : Integer; Legend : String)
5552:  Procedure TeX_SetCurvStep( CurvIndex, Step : Integer)
5553:  Procedure TeX_PlotCurve( X, Y : TVector; Lb, Ub, CurvIndex : Integer)
5554:  Procedure TeX_PlotCurveWithErrorBars( X, Y, S : TVector; Ns, Lb, Ub, CurvIndex : Integer)
5555:  Procedure TeX_PlotFunc( Func : TFunc; X1, X2 : Float; Npt : Integer; CurvIndex : Integer)
5556:  Procedure TeX_WriteLegend( NCurv : Integer; ShowPoints, ShowLines : Boolean)
5557:  Procedure TeX_ConRec( Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
5558:  Function Xcm( X : Float) : Float
5559:  Function Ycm( Y : Float) : Float
5560: end;
5561:
5562: *---------------------------------------------------------------------------*)
5563: procedure SIRegister_VarRecUtils(CL: TPSPascalCompiler);
5564: begin
5565:   TConstArray', 'array of TVarRec
5566:  Function CopyVarRec( const Item : TVarRec) : TVarRec
5567:  Function CreateConstArray( const Elements : array of const) : TConstArray
5568:  Procedure FinalizeVarRec( var Item : TVarRec)
5569:  Procedure FinalizeConstArray( var Arr : TConstArray)
```

```
5570: end;
5571:
5572: procedure SIRegister_StStrS(CL: TPSPascalCompiler);
5573: begin
5574:  Function HexBS( B : Byte) : ShortString
5575:  Function HexWS( W : Word) : ShortString
5576:  Function HexLS( L : LongInt) : ShortString
5577:  Function HexPtrS( P : Pointer) : ShortString
5578:  Function BinaryBS( B : Byte) : ShortString
5579:  Function BinaryWS( W : Word) : ShortString
5580:  Function BinaryLS( L : LongInt) : ShortString
5581:  Function OctalBS( B : Byte) : ShortString
5582:  Function OctalWS( W : Word) : ShortString
5583:  Function OctalLS( L : LongInt) : ShortString
5584:  Function Str2Int16S( const S : ShortString; var I : SmallInt) : Boolean
5585:  Function Str2WordS( const S : ShortString; var I : Word) : Boolean
5586:  Function Str2LongS( const S : ShortString; var I : LongInt) : Boolean
5587:  Function Str2RealS( const S : ShortString; var R : Double) : Boolean
5588:  Function Str2RealS( const S : ShortString; var R : Real) : Boolean
5589:  Function Str2ExtS( const S : ShortString; var R : Extended) : Boolean
5590:  Function Long2StrS( L : LongInt) : ShortString
5591:  Function Real2StrS( R : Double; Width : Byte; Places : ShortInt) : ShortString
5592:  Function Ext2StrS( R : Extended; Width : Byte; Places : ShortInt) : ShortString
5593:  Function ValPrepS( const S : ShortString) : ShortString
5594:  Function CharStrS( C : AnsiChar; Len : Cardinal) : ShortString
5595:  Function PadChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5596:  Function PadS( const S : ShortString; Len : Cardinal) : ShortString
5597:  Function LeftPadChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5598:  Function LeftPadS( const S : ShortString; Len : Cardinal) : ShortString
5599:  Function TrimLeadS( const S : ShortString) : ShortString
5600:  Function TrimTrailS( const S : ShortString) : ShortString
5601:  Function TrimS( const S : ShortString) : ShortString
5602:  Function TrimSpacesS( const S : ShortString) : ShortString
5603:  Function CenterChS( const S : ShortString; C : AnsiChar; Len : Cardinal) : ShortString
5604:  Function CenterS( const S : ShortString; Len : Cardinal) : ShortString
5605:  Function EntabS( const S : ShortString; TabSize : Byte) : ShortString
5606:  Function DetabS( const S : ShortString; TabSize : Byte) : ShortString
5607:  Function ScrambleS( const S, Key : ShortString) : ShortString
5608:  Function SubstituteS( const S, FromStr, ToStr : ShortString) : ShortString
5609:  Function FilterS( const S, Filters : ShortString) : ShortString
5610:  Function CharExistsS( const S : ShortString; C : AnsiChar) : Boolean
5611:  Function CharCountS( const S : ShortString; C : AnsiChar) : Byte
5612:  Function WordCountS( const S, WordDelims : ShortString) : Cardinal
5613:  Function WordPositionS( N : Cardinal; const S, WordDelims : ShortString; var Pos : Cardinal) : Boolean
5614:  Function ExtractWordS( N : Cardinal; const S, WordDelims : ShortString) : ShortString
5615:  Function AsciiCountS( const S, WordDelims : ShortString; Quote : AnsiChar) : Cardinal
5616:  Function AsciiPositionS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar;var Pos:Cardinal):Boolean
5617:  Function ExtractAsciiS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar): ShortString
5618:  Procedure WordWrapS(const InSt: ShortString; var OutSt,Overlap: ShortString;
      Margin:Cardinal;PadToMargin:Boolean)
5619:  Function CompStringS( const S1, S2 : ShortString) : Integer
5620:  Function CompUCStringS( const S1, S2 : ShortString) : Integer
5621:  Function SoundexS( const S : ShortString) : ShortString
5622:  Function MakeLetterSetS( const S : ShortString) : Longint
5623:  Procedure BMMakeTableS( const MatchString : ShortString; var BT : BTable)
5624:  Function BMSearchS(var Buffer,BufLength:Card;var BT:BTable;const MatchString:ShortString;var
      Pos:Cardinal):Bool;
5625:  Function BMSearchUCS(var Buffer,BufLength:Card;var BT:BTable;const MatchString:ShortString;var
      Pos:Cardinal):Bool;
5626:  Function DefaultExtensionS( const Name, Ext : ShortString) : ShortString
5627:  Function ForceExtensionS( const Name, Ext : ShortString) : ShortString
5628:  Function JustFilenameS( const PathName : ShortString) : ShortString
5629:  Function JustNameS( const PathName : ShortString) : ShortString
5630:  Function JustExtensionS( const Name : ShortString) : ShortString
5631:  Function JustPathnameS( const PathName : ShortString) : ShortString
5632:  Function AddBackSlashS( const DirName : ShortString) : ShortString
5633:  Function CleanPathNameS( const PathName : ShortString) : ShortString
5634:  Function HasExtensionS( const Name : ShortString; var DotPos : Cardinal) : Boolean
5635:  Function CommaizeS( L : LongInt) : ShortString
5636:  Function CommaizeChS( L : Longint; Ch : AnsiChar) : ShortString
5637:  Function FloatFormS(const Mask:ShortString;R:TstFloat;const LtCurr,RtCurr:ShortString;Sep,
      DecPt:Char):ShortString;
5638:  Function LongIntFormS(const Mask:ShortString;L:LongInt;const LtCurr,
      RtCurr:ShortString;Sep:AnsiChar):ShortString;
5639:  Function StrChPosS( const P : ShortString; C : AnsiChar; var Pos : Cardinal) : Boolean
5640:  Function StrStPosS( const P, S : ShortString; var Pos : Cardinal) : Boolean
5641:  Function StrStCopyS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5642:  Function StrChInsertS( const S : ShortString; C : AnsiChar; Pos : Cardinal) : ShortString
5643:  Function StrStInsertS( const S1, S2 : ShortString; Pos : Cardinal) : ShortString
5644:  Function StrChDeleteS( const S : ShortString; Pos : Cardinal) : ShortString
5645:  Function StrStDeleteS( const S : ShortString; Pos, Count : Cardinal) : ShortString
5646:  Function ContainsOnlyS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5647:  Function ContainsOtherThanS( const S, Chars : ShortString; var BadPos : Cardinal) : Boolean
5648:  Function CopyLeftS( const S : ShortString; Len : Cardinal) : ShortString
5649:  Function CopyMidS( const S : ShortString; First, Len : Cardinal) : ShortString
5650:  Function CopyRightS( const S : ShortString; First : Cardinal) : ShortString
5651:  Function CopyRightAbsS( const S : ShortString; NumChars : Cardinal) : ShortString
5652:  Function CopyFromNthWordS(const S,WordDelims:string;const AWord:String;N:Cardinal;var
      SubString:ShortString) :Boolean;
```

```
5653:  Function DeleteFromNthWordS(const S,WordDelims:String;AWord:ShortString;N:Cardinal;var
       SubStr:ShortString): Boolean;
5654:  Function CopyFromToWordS(const S,WordDelims,Word1,Word2:ShortString;N1,N2:Cardinal;var
       SubString:ShortString):Boolean;
5655:  Function DeleteFromToWordS(const S,WordDelims,Wrd1,Wrd2:ShortString;N1,N2:Cardinal;var
       SubString:ShortString):Boolean;
5656:  Function CopyWithinS( const S, Delimiter : ShortString; Strip : Boolean) : ShortString
5657:  Function DeleteWithinS( const S, Delimiter : ShortString) : ShortString
5658:  Function ExtractTokensS(const S,
       Delims:ShortString;QuoteChar:AnsiChar;AllowNulls:Boolean;Tokens:TStrings):Cardinal
5659:  Function IsChAlphaS( C : Char) : Boolean
5660:  Function IsChNumericS( C : Char; const Numbers : ShortString) : Boolean
5661:  Function IsChAlphaNumericS( C : Char; const Numbers : ShortString) : Boolean
5662:  Function IsStrAlphaS( const S : Shortstring) : Boolean
5663:  Function IsStrNumericS( const S, Numbers : ShortString) : Boolean
5664:  Function IsStrAlphaNumericS( const S, Numbers : ShortString) : Boolean
5665:  Function LastWordS( const S, WordDelims, AWord : ShortString; var Position : Cardinal) : Boolean
5666:  Function LastWordAbsS( const S, WordDelims : ShortString; var Position : Cardinal) : Boolean
5667:  Function LastStringS( const S, AString : ShortString; var Position : Cardinal) : Boolean
5668:  Function LeftTrimCharsS( const S, Chars : ShortString) : ShortString
5669:  Function KeepCharsS( const S, Chars : ShortString) : ShortString
5670:  Function RepeatStringS(const RepeatString:ShortString;var Repetitions: Cardinal; MaxLen :
       Cardinal):ShortString;
5671:  Function ReplaceStringS(const S,OldString,NewString:ShortString;N:Cardinal;var
       Replacements:Cardinal):ShortString;
5672:  Function ReplaceStringAllS(const S,OldString,NewString:ShortString;var Replacements:Cardinal):ShortString;
5673:  Function ReplaceWordS(const S,WordDelims,OldWord,NewW:ShortString;N:Cardinal;var
       Replacements:Cardinal):ShortString
5674:  Function ReplaceWordAllS(const S,WordDelims,OldWord,NewWord:ShortString;var
       Replacements:Cardinal):ShortString
5675:  Function RightTrimCharsS( const S, Chars : ShortString) : ShortString
5676:  Function StrWithinS(const S,SearchStr: ShortString;Start:Cardinal;var Position:Cardinal):boolean
5677:  Function TrimCharsS( const S, Chars : ShortString) : ShortString
5678:  Function WordPosS(const S,WordDelims,AWord:ShortString;N:Cardinal; var Position: Cardinal):Boolean
5679: end;
5680:
5681:
5682: ********unit uPSI_StUtils; from Systools4*************************************************************
5683: Function SignL( L : LongInt) : Integer
5684: Function SignF( F : Extended) : Integer
5685: Function MinWord( A, B : Word) : Word
5686: Function MidWord( W1, W2, W3 : Word) : Word
5687: Function MaxWord( A, B : Word) : Word
5688: Function MinLong( A, B : LongInt) : LongInt
5689: Function MidLong( L1, L2, L3 : LongInt) : LongInt
5690: Function MaxLong( A, B : LongInt) : LongInt
5691: Function MinFloat( F1, F2 : Extended) : Extended
5692: Function MidFloat( F1, F2, F3 : Extended) : Extended
5693: Function MaxFloat( F1, F2 : Extended) : Extended
5694: Function MakeInteger16( H, L : Byte) : SmallInt
5695: Function MakeWordS( H, L : Byte) : Word
5696: Function SwapNibble( B : Byte) : Byte
5697: Function SwapWord( L : LongInt) : LongInt
5698: Procedure SetFlag( var Flags : Word; FlagMask : Word)
5699: Procedure ClearFlag( var Flags : Word; FlagMask : Word)
5700: Function FlagIsSet( Flags, FlagMask : Word) : Boolean
5701: Procedure SetByteFlag( var Flags : Byte; FlagMask : Byte)
5702: Procedure ClearByteFlag( var Flags : Byte; FlagMask : Byte)
5703: Function ByteFlagIsSet( Flags, FlagMask : Byte) : Boolean
5704: Procedure SetLongFlag( var Flags : LongInt; FlagMask : LongInt)
5705: Procedure ClearLongFlag( var Flags : LongInt; FlagMask : LongInt)
5706: Function LongFlagIsSet( Flags, FlagMask : LongInt) : Boolean
5707: Procedure ExchangeBytes( var I, J : Byte)
5708: Procedure ExchangeWords( var I, J : Word)
5709: Procedure ExchangeLongInts( var I, J : LongInt)
5710: Procedure ExchangeStructs( var I, J, Size : Cardinal)
5711: Procedure FillWord( var Dest, Count : Cardinal; Filler : Word)
5712: Procedure FillStruct( var Dest, Count : Cardinal; var Filler, FillerSize : Cardinal)
5713: Function AddWordToPtr( P : ___Pointer; W : Word) : ___Pointer
5714: //******************uPSI_StFIN;************************************************************
5715: Function AccruedInterestMaturity(Issue,Maturity:TStDate;Rate,Par:Extended;Basis: TStBasis): Extended
5716: Function AccruedInterestPeriodic(Issue,Settlement,Maturity:TStDate;Rate,
       Par:Extended;Frequency:TStFrequency; Basis : TStBasis) : Extended
5717: Function BondDuration( Settlement,Maturity:TStDate;Rate,
       Yield:Extended;Frequency:TStFrequency;Basis:TStBasis):Extended;
5718: Function BondPrice(Settlement,Maturity:TStDate;Rate,Yield,
       Redemption:Extended;Freq:TStFrequency;Basis:TStBasis): Extended
5719: Function CumulativeInterest( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
       Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended
5720: Function CumulativePrincipal( Rate : Extended; NPeriods : Integer; PV : Extended; StartPeriod, EndPeriod :
       Integer; Frequency : TStFrequency; Timing : TStPaymentTime) : Extended
5721: Function DayCount( Day1, Day2 : TStDate; Basis : TStBasis) : LongInt
5722: Function DecliningBalance( Cost, Salvage : Extended; Life, Period, Month : Integer) : Extended
5723: Function DiscountRate(Settlement,Maturity:TStDate; Price,Redemption:Extended;Basis:TStBasis): Extended;
5724: Function DollarToDecimal( FracDollar : Extended; Fraction : Integer) : Extended
5725: Function DollarToDecimalText( DecDollar : Extended) : string
5726: Function DollarToFraction( DecDollar : Extended; Fraction : Integer) : Extended
5727: Function DollarToFractionStr( FracDollar : Extended; Fraction : Integer) : string
5728: Function EffectiveInterestRate( NominalRate : Extended; Frequency : TStFrequency) : Extended
```

```
5729: Function FutureValueS(Rate:Extended;NPeriods:Int;Pmt,
      PV:Extended;Freq:TStFrequency;Timing:TStPaymentTime):Extended;
5730: Function FutureValueSchedule( Principal : Extended; const Schedule : array of Double) : Extended
5731: Function FutureValueSchedule16( Principal : Extended; const Schedule, NRates : Integer) : Extended
5732: Function InterestRateS(NPeriods:Int;Pmt,PV,
      FV:Extended;Freq:TStFrequency;Timing:TStPaymentTime;Guess:Extended):Extend;
5733: Function InternalRateOfReturn( const Values : array of Double; Guess : Extended) : Extended
5734: Function InternalRateOfReturn16( const Values, NValues : Integer; Guess : Extended) : Extended
5735: Function IsCardValid( const S : string) : Boolean
5736: Function ModifiedDuration(Settlement,Maturity:TStDate;Rate,
      Yield:Extended;Freq:TStFrequency;Basis:TStBasis):Extended;
5737: Function ModifiedIRR(const Values: array of Double; FinanceRate,ReinvestRate: Extended) : Extended
5738: Function ModifiedIRR16(const Values,NValues:Integer; FinanceRate,ReinvestRate: Extended) : Extended
5739: Function NetPresentValueS( Rate : Extended; const Values : array of Double) : Extended
5740: Function NetPresentValue16( Rate : Extended; const Values, NValues : Integer) : Extended
5741: Function NominalInterestRate( EffectRate : Extended; Frequency : TStFrequency) : Extended
5742: Function NonperiodicIRR(const Values:array of Double;const Dates:array of TStDate;Guess:Extended):Extended;
5743: Function NonperiodicNPV(Rate:Extended;const Values: array of Double;const Dates:array of TStDate):Extended;
5744: Function Payment( Rate : Extended; NPeriods : Integer; PV, FV : Extended; Frequency : TStFrequency; Timing
      : TStPaymentTime): Extended
5745: Function Periods(Rate:Extended;Pmt,PV,FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Integer;
5746: Function PresentValueS( Rate: Extended; NPeriods: Integer; Pmt, FV: Extended; Frequency: TStFrequency;
      Timing: TStPaymentTime): Extended
5747: Function ReceivedAtMaturity(Settlement,Maturity:TStDate;Invest,Discount:Extended;Basis:TStBasis):Extended;
5748: Function RoundToDecimal( Value : Extended; Places : Integer; Bankers : Boolean) : Extended
5749: Function TBillEquivYield( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5750: Function TBillPrice( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5751: Function TBillYield( Settlement, Maturity : TStDate; Price : Extended) : Extended
5752: Function VariableDecliningBalance( Cost, Salvage : Extended; Life : Integer; StartPeriod, EndPeriod,
      Factor : Extended; NoSwitch : boolean) : Extended
5753: Function YieldDiscounted(Settlement,Maturity:TStDate;Price,Redemption:Extended;Basis:TStBasis):Extended;
5754: Function YieldPeriodic(Settlement,Maturity:TStDate;Rate,Price,
      Redemption:Extended;Freq:TStFrequency;Basis:TStBasis): Extended
5755: Function YieldMaturity(Issue,Settlement,Maturity:TStDate;Rate,Price:Extended;Basis:TStBasis):Extended;
5756:
5757: //*****************************************unit uPSI_StAstroP;
5758: Procedure PlanetsPos( JD : Double; var PA : TStPlanetsArray)
5759: //*****unit unit uPSI_StStat; Statistic Package of SysTools***************************************
5760: Function AveDev( const Data : array of Double) : Double
5761: Function AveDev16( const Data, NData : Integer) : Double
5762: Function Confidence( Alpha, StandardDev : Double; Size : LongInt) : Double
5763: Function Correlation( const Data1, Data2 : array of Double) : Double
5764: Function Correlation16( const Data1, Data2, NData : Integer) : Double
5765: Function Covariance( const Data1, Data2 : array of Double) : Double
5766: Function Covariance16( const Data1, Data2, NData : Integer) : Double
5767: Function DevSq( const Data : array of Double) : Double
5768: Function DevSq16( const Data, NData : Integer) : Double
5769: Procedure Frequency(const Data:array of Double;const Bins:array of Double;var Counts: array of LongInt);
5770:  //Procedure Frequency16( const Data, NData : Integer; const Bins, NBins : Integer; var Counts)
5771: Function GeometricMeanS( const Data : array of Double) : Double
5772: Function GeometricMean16( const Data, NData : Integer) : Double
5773: Function HarmonicMeanS( const Data : array of Double) : Double
5774: Function HarmonicMean16( const Data, NData : Integer) : Double
5775: Function Largest( const Data : array of Double; K : Integer) : Double
5776: Function Largest16( const Data, NData : Integer; K : Integer) : Double
5777: Function MedianS( const Data : array of Double) : Double
5778: Function Median16( const Data, NData : Integer) : Double
5779: Function Mode( const Data : array of Double) : Double
5780: Function Mode16( const Data, NData : Integer) : Double
5781: Function Percentile( const Data : array of Double; K : Double) : Double
5782: Function Percentile16( const Data, NData : Integer; K : Double) : Double
5783: Function PercentRank( const Data : array of Double; X : Double) : Double
5784: Function PercentRank16( const Data, NData : Integer; X : Double) : Double
5785: Function Permutations( Number, NumberChosen : Integer) : Extended
5786: Function Combinations( Number, NumberChosen : Integer) : Extended
5787: Function FactorialS( N : Integer) : Extended
5788: Function Rank( Number : Double; const Data : array of Double; Ascending : Boolean) : Integer
5789: Function Rank16( Number : Double; const Data, NData : Integer; Ascending : Boolean) : Integer
5790: Function Smallest( const Data : array of Double; K : Integer) : Double
5791: Function Smallest16( const Data, NData : Integer; K : Integer) : Double
5792: Function TrimMean( const Data : array of Double; Percent : Double) : Double
5793: Function TrimMean16( const Data, NData : Integer; Percent : Double) : Double
5794:   AddTypeS('TStLinEst', 'record B0 : Double; B1 : double; seB0 : double; seB'
5795:    +'1 : Double; R2 : Double; sigma :Double; SSr: double; SSe: Double; F0 : Double; df : Integer;end
5796: Procedure LinEst(const KnownY:array of Double;const KnownX:array of Double;var
      LF:TStLinEst;ErrorStats:Bool;
5797: Procedure LogEst(const KnownY:array of Double;const KnownX:array of Double;var
      LF:TStLinEst;ErrorStats:Bool;
5798: Function Forecast(X: Double; const KnownY: array of Double; const KnownX: array of Double) : Double
5799: Function ForecastExponential(X:Double;const KnownY:array of Double;const KnownX:array of Double):Double;
5800: Function Intercept( const KnownY : array of Double; const KnownX : array of Double) : Double
5801: Function RSquared( const KnownY : array of Double; const KnownX : array of Double) : Double
5802: Function Slope( const KnownY : array of Double; const KnownX : array of Double) : Double
5803: Function StandardErrorY( const KnownY : array of Double; const KnownX : array of Double) : Double
5804: Function BetaDist( X, Alpha, Beta, A, B : Single) : Single
5805: Function BetaInv( Probability, Alpha, Beta, A, B : Single) : Single
5806: Function BinomDist( NumberS, Trials : Integer; ProbabilityS : Single; Cumulative : Boolean) : Single
5807: Function CritBinom( Trials : Integer; ProbabilityS, Alpha : Single) : Integer
5808: Function ChiDist( X : Single; DegreesFreedom : Integer) : Single
```

```
5809: Function ChiInv( Probability : Single; DegreesFreedom : Integer) : Single
5810: Function ExponDist( X, Lambda : Single; Cumulative : Boolean) : Single
5811: Function FDist( X : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5812: Function FInv( Probability : Single; DegreesFreedom1, DegreesFreedom2 : Integer) : Single
5813: Function LogNormDist( X, Mean, StandardDev : Single) : Single
5814: Function LogInv( Probability, Mean, StandardDev : Single) : Single
5815: Function NormDist( X, Mean, StandardDev : Single; Cumulative : Boolean) : Single
5816: Function NormInv( Probability, Mean, StandardDev : Single) : Single
5817: Function NormSDist( Z : Single) : Single
5818: Function NormSInv( Probability : Single) : Single
5819: Function Poisson( X : Integer; Mean : Single; Cumulative : Boolean) : Single
5820: Function TDist( X : Single; DegreesFreedom : Integer; TwoTails : Boolean) : Single
5821: Function TInv( Probability : Single; DegreesFreedom : Integer) : Single
5822: Function Erfc( X : Single) : Single
5823: Function GammaLn( X : Single) : Single
5824: Function LargestSort( const Data : array of Double; K : Integer) : Double
5825: Function SmallestSort( const Data : array of double; K : Integer) : Double
5826:
5827: procedure SIRegister_TStSorter(CL: TPSPascalCompiler);
5828:  Function OptimumHeapToUse( RecLen : Cardinal; NumRecs : LongInt) : LongInt
5829:  Function MinimumHeapToUse( RecLen : Cardinal) : LongInt
5830:  Function MergeInfo( MaxHeap : LongInt; RecLen : Cardinal; NumRecs : LongInt) : TMergeInfo
5831:  Function DefaultMergeName( MergeNum : Integer) : string
5832:  Procedure ArraySort( var A, RecLen, NumRecs : Cardinal; Compare : TUntypedCompareFunc)
5833:
5834: procedure SIRegister_StAstro(CL: TPSPascalCompiler);
5835: Function AmountOfSunlight( LD : TStDate; Longitude, Latitude : Double) : TStTime
5836:  Function FixedRiseSet( LD : TStDate; RA, DC, Longitude, Latitude : Double) : TStRiseSetRec
5837:  Function SunPos( UT : TStDateTimeRec) : TStPosRec
5838:  Function SunPosPrim( UT : TStDateTimeRec) : TStSunXYZRec
5839:  Function SunRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5840:  Function Twilight( LD : TStDate; Longitude, Latitude : Double; TwiType:TStTwilight):TStRiseSetRec
5841:  Function LunarPhase( UT : TStDateTimeRec) : Double
5842:  Function MoonPos( UT : TStDateTimeRec) : TStMoonPosRec
5843:  Function MoonRiseSet( LD : TStDate; Longitude, Latitude : Double) : TStRiseSetRec
5844:  Function FirstQuarter( D : TStDate) : TStLunarRecord
5845:  Function FullMoon( D : TStDate) : TStLunarRecord
5846:  Function LastQuarter( D : TStDate) : TStLunarRecord
5847:  Function NewMoon( D : TStDate) : TStLunarRecord
5848:  Function NextFirstQuarter( D : TStDate) : TStDateTimeRec
5849:  Function NextFullMoon( D : TStDate) : TStDateTimeRec
5850:  Function NextLastQuarter( D : TStDate) : TStDateTimeRec
5851:  Function NextNewMoon( D : TStDate) : TStDateTimeRec
5852:  Function PrevFirstQuarter( D : TStDate) : TStDateTimeRec
5853:  Function PrevFullMoon( D : TStDate) : TStDateTimeRec
5854:  Function PrevLastQuarter( D : TStDate) : TStDateTimeRec
5855:  Function PrevNewMoon( D : TStDate) : TStDateTimeRec
5856:  Function SiderealTime( UT : TStDateTimeRec) : Double
5857:  Function Solstice( Y, Epoch : Integer; Summer : Boolean) : TStDateTimeRec
5858:  Function Equinox( Y, Epoch : Integer; Vernal : Boolean) : TStDateTimeRec
5859:  Function SEaster( Y, Epoch : Integer) : TStDate
5860:  Function DateTimeToAJD( D : TDateTime) : Double
5861:  Function HoursMin( RA : Double) : ShortString
5862:  Function DegsMin( DC : Double) : ShortString
5863:  Function AJDToDateTime( D : Double) : TDateTime
5864:
5865: Procedure SIRegister_StDate(CL: TPSPascalCompiler);
5866:  Function CurrentDate : TStDate
5867:  Function StValidDate( Day, Month, Year, Epoch : Integer) : Boolean
5868:  Function DMYtoStDate( Day, Month, Year, Epoch : Integer) : TStDate
5869:  Procedure StDateToDMY( Julian : TStDate; var Day, Month, Year : Integer)
5870:  Function StIncDate( Julian : TStDate; Days, Months, Years : Integer) : TStDate
5871:  Function IncDateTrunc( Julian : TStDate; Months, Years : Integer) : TStDate
5872:  Procedure StDateDiff( Date1, Date2 : TStDate; var Days, Months, Years : Integer)
5873:  Function BondDateDiff( Date1, Date2 : TStDate; DayBasis : TStBondDateType) : TStDate
5874:  Function WeekOfYear( Julian : TStDate) : Byte
5875:  Function AstJulianDate( Julian : TStDate) : Double
5876:  Function AstJulianDatetoStDate( AstJulian : Double; Truncate : Boolean) : TStDate
5877:  Function AstJulianDatePrim( Year, Month, Date : Integer; UT : TStTime) : Double
5878:  Function StDayOfWeek( Julian : TStDate) : TStDayType
5879:  Function DayOfWeekDMY( Day, Month, Year, Epoch : Integer) : TStDayType
5880:  Function StIsLeapYear( Year : Integer) : Boolean
5881:  Function StDaysInMonth( Month : Integer; Year, Epoch : Integer) : Integer
5882:  Function ResolveEpoch( Year, Epoch : Integer) : Integer
5883:  Function ValidTime( Hours, Minutes, Seconds : Integer) : Boolean
5884:  Procedure StTimeToHMS( T : TStTime; var Hours, Minutes, Seconds : Byte)
5885:  Function HMStoStTime( Hours, Minutes, Seconds : Byte) : TStTime
5886:  Function CurrentTime : TStTime
5887:  Procedure TimeDiff( Time1, Time2 : TStTime; var Hours, Minutes, Seconds : Byte)
5888:  Function StIncTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5889:  Function DecTime( T : TStTime; Hours, Minutes, Seconds : Byte) : TStTime
5890:  Function RoundToNearestHour( T : TStTime; Truncate : Boolean) : TStTime
5891:  Function RoundToNearestMinute( const T : TStTime; Truncate : Boolean) : TStTime
5892:  Procedure DateTimeDiff(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;var Days:LongInt;var Secs:LongInt
5893:  Procedure IncDateTime(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;Days:Integer;Secs:LongInt)
5894:  Function DateTimeToStDate( DT : TDateTime) : TStDate
5895:  Function DateTimeToStTime( DT : TDateTime) : TStTime
5896:  Function StDateToDateTime( D : TStDate) : TDateTime
5897:  Function StTimeToDateTime( T : TStTime) : TDateTime
```

```
5898:  Function Convert2ByteDate( TwoByteDate : Word) : TStDate
5899:  Function Convert4ByteDate( FourByteDate : TStDate) : Word
5900:
5901: Procedure SIRegister_StDateSt(CL: TPSPascalCompiler);
5902: Function DateStringHMStoAstJD( const Picture, DS : string; H, M, S, Epoch : integer) : Double
5903:  Function MonthToString( const Month : Integer) : string
5904:  Function DateStringToStDate( const Picture, S : string; Epoch : Integer) : TStDate
5905:  Function DateStringToDMY(const Picture,S:string; Epoch:Integer; var D, M, Y : Integer):Boolean
5906:  Function StDateToDateString( const Picture : string; const Julian : TStDate; Pack : Boolean):string
5907:  Function DayOfWeekToString( const WeekDay : TStDayType) : string
5908:  Function DMYtoDateString(const Picture:string;Day,Month,Year,Epoch:Integer;Pack:Boolean): string);
5909:  Function CurrentDateString( const Picture : string; Pack : Boolean) : string
5910:  Function CurrentTimeString( const Picture : string; Pack : Boolean) : string
5911:  Function TimeStringToHMS( const Picture, St : string; var H, M, S : Integer) : Boolean
5912:  Function TimeStringToStTime( const Picture, S : string) : TStTime
5913:  Function StTimeToAmPmString( const Picture : string; const T : TStTime; Pack : Boolean) : string
5914:  Function StTimeToTimeString( const Picture : string; const T : TStTime; Pack : Boolean) : string
5915:  Function DateStringIsBlank( const Picture, S : string) : Boolean
5916:  Function InternationalDate( ForceCentury : Boolean) : string
5917:  Function InternationalLongDate( ShortNames : Boolean; ExcludeDOW : Boolean) : string
5918:  Function InternationalTime( ShowSeconds : Boolean) : string
5919:  Procedure ResetInternationalInfo
5920:
5921: procedure SIRegister_StBase(CL: TPSPascalCompiler);
5922:  Function DestroyNode( Container : TStContainer; Node : TStNode; OtherData : Pointer) : Boolean
5923:  Function AnsiUpperCaseShort32( const S : string) : string
5924:  Function AnsiCompareTextShort32( const S1, S2 : string) : Integer
5925:  Function AnsiCompareStrShort32( const S1, S2 : string) : Integer
5926:  Function HugeCompressRLE( const InBuffer, InLen : Longint; var OutBuffer) : Longint
5927:  Function HugeDecompressRLE( const InBuffer, InLen : Longint; var OutBuffer,OutLen:LongInt): Longint
5928:  Procedure HugeFillChar( var Dest, Count : Longint; Value : Byte)
5929:  Procedure HugeFillStruc( var Dest, Count : Longint; const Value, ValSize : Cardinal)
5930:  Function Upcase( C : AnsiChar) : AnsiChar
5931:  Function LoCase( C : AnsiChar) : AnsiChar
5932:  Function CompareLetterSets( Set1, Set2 : LongInt) : Cardinal
5933:  Function CompStruct( const S1, S2, Size : Cardinal) : Integer
5934:  Function Search(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardinal):Bool;
5935:  Function StSearch(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
5936:  Function SearchUC(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
5937:  Function IsOrInheritsFrom( Root, Candidate : TClass) : boolean
5938:  Procedure RaiseContainerError( Code : longint)
5939:  Procedure RaiseContainerErrorFmt( Code : Longint; Data : array of const)
5940:  Function ProductOverflow( A, B : LongInt) : Boolean
5941:  Function StNewStr( S : string) : PShortString
5942:  Procedure StDisposeStr( PS : PShortString)
5943:  Procedure ValLongInt( S : ShortString; var LI : Longint; var ErrorCode : integer)
5944:  Procedure ValSmallint( const S : ShortString; var SI : smallint; var ErrorCode : integer)
5945:  Procedure ValWord( const S : ShortString; var Wd : word; var ErrorCode : integer)
5946:  Procedure RaiseStError( ExceptionClass : EStExceptionClass; Code : LongInt)
5947:  Procedure RaiseStWin32Error( ExceptionClass : EStExceptionClass; Code : LongInt)
5948:  Procedure RaiseStWin32ErrorEx( ExceptionClass : EStExceptionClass; Code : LongInt; Info : string)
5949:
5950: procedure SIRegister_usvd(CL: TPSPascalCompiler);
5951: begin
5952:  Procedure SV_Decomp( A : TMatrix; Lb, Ub1, Ub2 : Integer; S : TVector; V : TMatrix)
5953:  Procedure SV_SetZero( S : TVector; Lb, Ub : Integer; Tol : Float)
5954:  Procedure SV_Solve(U:TMatrix; S:TVector;V:TMatrix;B:TVector;Lb,Ub1,Ub2:Integer;X:TVector);
5955:  Procedure SV_Approx( U : TMatrix; S : TVector; V : TMatrix; Lb, Ub1, Ub2 : Integer; A : TMatrix)
5956:  Procedure RKF45(F:TDiffEqs;Neqn:Int;Y,Yp:TVector;var T:Float;Tout,RelErr,AbsErr:Float;var Flag:Int;
5957: end;
5958:
5959: //**********unit unit ; StMath Package of SysTools*****************************************
5960: Function IntPowerS( Base : Extended; Exponent : Integer) : Extended
5961: Function PowerS( Base, Exponent : Extended) : Extended
5962: Function StInvCos( X : Double) : Double
5963: Function StInvSin( Y : Double) : Double
5964: Function StInvTan2( X, Y : Double) : Double
5965: Function StTan( A : Double) : Double
5966: Procedure DumpException;  //unit StExpEng;
5967: Function HexifyBlock( var Buffer, BufferSize : Integer) : string
5968:
5969: //**********unit unit ; StCRC Package of SysTools*****************************************
5970: Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
5971: Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
5972: Function Adler32OfFile( FileName : AnsiString) : LongInt
5973: Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
5974: Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
5975: Function Crc16OfFile( FileName : AnsiString) : Cardinal
5976: Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
5977: Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
5978: Function Crc32OfFile( FileName : AnsiString) : LongInt
5979: Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
5980: Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
5981: Function InternetSumOfFile( FileName : AnsiString) : Cardinal
5982: Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
5983: Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
5984: Function Kermit16OfFile( FileName : AnsiString) : Cardinal
5985:
5986: //**********unit unit ; StBCD Package of SysTools*****************************************
```

```
5987: Function AddBcd( const B1, B2 : TbcdS) : TbcdS
5988: Function SubBcd( const B1, B2 : TbcdS) : TbcdS
5989: Function MulBcd( const B1, B2 : TbcdS) : TbcdS
5990: Function DivBcd( const B1, B2 : TbcdS) : TbcdS
5991: Function ModBcd( const B1, B2 : TbcdS) : TbcdS
5992: Function NegBcd( const B : TbcdS) : TbcdS
5993: Function AbsBcd( const B : TbcdS) : TbcdS
5994: Function FracBcd( const B : TbcdS) : TbcdS
5995: Function IntBcd( const B : TbcdS) : TbcdS
5996: Function RoundDigitsBcd( const B : TbcdS; Digits : Cardinal) : TbcdS
5997: Function RoundPlacesBcd( const B : TbcdS; Places : Cardinal) : TbcdS
5998: Function ValBcd( const S : string) : TbcdS
5999: Function LongBcd( L : LongInt) : TbcdS
6000: Function ExtBcd( E : Extended) : TbcdS
6001: Function ExpBcd( const B : TbcdS) : TbcdS
6002: Function LnBcd( const B : TbcdS) : TbcdS
6003: Function IntPowBcd( const B : TbcdS; E : LongInt) : TbcdS
6004: Function PowBcd( const B, E : TbcdS) : TbcdS
6005: Function SqrtBcd( const B : TbcdS) : TbcdS
6006: Function CmpBcd( const B1, B2 : TbcdS) : Integer
6007: Function EqDigitsBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6008: Function EqPlacesBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6009: Function IsIntBcd( const B : TbcdS) : Boolean
6010: Function TruncBcd( const B : TbcdS) : LongInt
6011: Function BcdExt( const B : TbcdS) : Extended
6012: Function RoundBcd( const B : TbcdS) : LongInt
6013: Function StrBcd( const B : TbcdS; Width, Places : Cardinal) : string
6014: Function StrExpBcd( const B : TbcdS; Width : Cardinal) : string
6015: Function FormatBcd( const Format : string; const B : TbcdS) : string
6016: Function StrGeneralBcd( const B : TbcdS) : string
6017: Function FloatFormBcd(const Mask:string;B:TbcdS;const LtCurr,RtCurr:string;Sep,DecPt:AnsiChar):string
6018: Procedure ConvertBcd( const SrcB, SrcSize : Byte; var DestB, DestSize : Byte)
6019:
6020: ////*******unit unit ; StTxtDat; TStTextDataRecordSet Package of SysTools**********************
6021: Procedure StParseLine( const Data : AnsiString; Schema : TStTextDataSchema; Result : TStrings)
6022: Function StFieldTypeToStr( FieldType : TStSchemaFieldType) : AnsiString
6023: Function StStrToFieldType( const S : AnsiString) : TStSchemaFieldType
6024: Function StDeEscape( const EscStr : AnsiString) : Char
6025: Function StDoEscape( Delim : Char) : AnsiString
6026: Function StTrimTrailingChars( const S : AnsiString; Trailer : Char) : AnsiString
6027: Function AnsiHashText( const S : string; Size : Integer) : Integer
6028: Function AnsiHashStr( const S : string; Size : Integer) : Integer
6029: Function AnsiELFHashText( const S : string; Size : Integer) : Integer
6030: Function AnsiELFHashStr( const S : string; Size : Integer) : Integer
6031:
6032: //**********unit unit ; StNetCon Package of SysTools*****************************************
6033:   with AddClassN(FindClass('TStComponent'),'TStNetConnection') do begin
6034:     Constructor Create( AOwner : TComponent)
6035:     Function Connect : DWord
6036:     Function Disconnect : DWord
6037:     RegisterProperty('Password', 'String', iptrw);
6038:     Property('UserName', 'String', iptrw);
6039:     Property('ConnectOptions', 'TStNetConnectOptionsSet', iptrw);
6040:     Property('DisconnectOptions', 'TStNetDisconnectOptionsSet', iptrw);
6041:     Property('LocalDevice', 'String', iptrw);
6042:     Property('ServerName', 'String', iptrw);
6043:     Property('ShareName', 'String', iptrw);
6044:     Property('OnConnect', 'TNotifyEvent', iptrw);
6045:     Property('OnConnectFail', 'TOnConnectFailEvent', iptrw);
6046:     Property('OnConnectCancel', 'TOnConnectCancelEvent', iptrw);
6047:     Property('OnDisconnect', 'TNotifyEvent', iptrw);
6048:     Property('OnDisconnectFail', 'TOnDisconnectFailEvent', iptrw);
6049:     Property('OnDisconnectCancel', 'TOnDisconnectCancelEvent', iptrw);
6050:   end;
6051: //***********Thread Functions Context of Win API --- more objects in SyncObjs.pas
6052: / /153 unit uPSI_SyncObjs, unit uPSIParallelJobs;
6053: Procedure InitializeCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6054: Procedure EnterCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6055: Procedure LeaveCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6056: Function InitializeCriticalSectionAndSpinCount(var
     lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):BOOL;
6057: Function SetCriticalSectionSpinCount(var lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):DWORD;
6058: Function TryEnterCriticalSection( var lpCriticalSection : TRTLCriticalSection) : BOOL
6059: Procedure DeleteCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6060: Function GetThreadContext( hThread : THandle; var lpContext : TContext) : BOOL
6061: Function SetThreadContext( hThread : THandle; const lpContext : TContext) : BOOL
6062: Function SuspendThread( hThread : THandle) : DWORD
6063: Function ResumeThread( hThread : THandle) : DWORD
6064: Function CreateThread2(ThreadFunc: TThreadFunction2; thrid: DWord) : THandle
6065: Function GetCurrentThread : THandle
6066: Procedure ExitThread( dwExitCode : DWORD)
6067: Function TerminateThread( hThread : THandle; dwExitCode : DWORD) : BOOL
6068: Function GetExitCodeThread( hThread : THandle; var lpExitCode : DWORD) : BOOL
6069: Procedure EndThread(ExitCode: Integer);
6070: Function WaitForSingleObject( hHandle : THandle; dwMilliseconds : DWORD) : DWORD
6071: Function MakeProcInstance( Proc : FARPROC; Instance : THandle) : FARPROC
6072: Procedure FreeProcInstance( Proc : FARPROC)
6073: Procedure FreeLibraryAndExitThread( hLibModule : HMODULE; dwExitCode : DWORD)
6074: Function DisableThreadLibraryCalls( hLibModule : HMODULE) : BOOL
```

```
6075: Procedure ParallelJob( ASelf : TObject; ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6076: Procedure ParallelJob1( ATarget : Pointer; AParam : Pointer; ASafeSection : boolean);
6077: Procedure
      ParallelJob2(AJobGroup:TJobsGroup;ASelf:TObject;ATarget:Pointer;AParam:Pointer;ASafeSection:boolean);
6078: Procedure ParallelJob3( AJobGroup:TJobsGroup;ATarget:Pointer;AParam:Pointer;ASafeSection: boolean);
6079: Function CreateParallelJob(ASelf:TObject;ATarget:Pointer;AParam:Pointer;ASafeSection:boolean):TParallelJob;
6080: Function CreateParallelJob1(ATarget:Pointer; AParam:Pointer; ASafeSection : boolean) : TParallelJob;
6081: Function CurrentParallelJobInfo : TParallelJobInfo
6082: Function ObtainParallelJobInfo : TParallelJobInfo
6083:
6084: ************************************************unit uPSI_JclMime;
6085:  Function MimeEncodeString( const S : AnsiString) : AnsiString
6086:  Function MimeDecodeString( const S : AnsiString) : AnsiString
6087:  Procedure MimeEncodeStream( const InputStream : TStream; const OutputStream : TStream)
6088:  Procedure MimeDecodeStream( const InputStream : TStream; const OutputStream : TStream)
6089:  Function MimeEncodedSize( const I : Cardinal) : Cardinal
6090:  Function MimeDecodedSize( const I : Cardinal) : Cardinal
6091:  Procedure MimeEncode( var InputBuffer : string; const InputByteCount : Cardinal; var OutputBuffer)
6092:  Function MimeDecode(var InputBuffer:string;const InputBytesCount:Cardinal;var OutputBuffer):Cardinal;
6093:  Function MimeDecodePartial(var InputBuffer:string;const InputBytesCount:Cardinal;var
      OutputBuffer:string;var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : Cardinal
6094:  Function MimeDecodePartialEnd(var OutputBuf:string;const ByteBuf:Cardinal;const
      ByteBufferSpace:Cardinal): Cardinal;
6095:
6096: ************************************************unit uPSI_JclPrint;
6097:  Procedure DirectPrint( const Printer, Data : string)
6098:  Procedure SetPrinterPixelsPerInch
6099:  Function GetPrinterResolution : TPoint
6100:  Function CharFitsWithinDots( const Text : string; const Dots : Integer) : Integer
6101:  Procedure PrintMemo( const Memo : TMemo; const Rect : TRect)
6102:
6103:
6104: //*************************************unit uPSI_ShLwApi;************************************
6105:  Function StrChr( lpStart : PChar; wMatch : WORD) : PChar
6106:  Function StrChrI( lpStart : PChar; wMatch : WORD) : PChar
6107:  Function StrCmpN( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6108:  Function StrCmpNI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6109:  Function StrCSpn( lpStr_, lpSet : PChar) : Integer
6110:  Function StrCSpnI( lpStr1, lpSet : PChar) : Integer
6111:  Function StrDup( lpSrch : PChar) : PChar
6112:  Function StrFormatByteSize( dw : DWORD; szBuf : PChar; uiBufSize : UINT) : PChar
6113:  Function StrFormatKBSize( qdw : Dword; szBuf : PChar; uiBufSize : UINT) : PChar
6114:  Function StrFromTimeInterval(pszOut: PChar; cchMax:UINT;dwTimeMS:DWORD; digits: Integer): Integer
6115:  Function StrIsIntlEqual( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6116: /Function StrNCat( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6117:  Function StrPBrk( psz, pszSet : PChar) : PChar
6118:  Function StrRChr( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6119:  Function StrRChrI( lpStart, lpEnd : PChar; wMatch : WORD) : PChar
6120:  Function StrRStrI( lpSource, lpLast, lpSrch : PChar) : PChar
6121:  Function StrSpn( psz, pszSet : PChar) : Integer
6122:  Function StrStr( lpFirst, lpSrch : PChar) : PChar
6123:  Function StrStrI( lpFirst, lpSrch : PChar) : PChar
6124:  Function StrToInt( lpSrch : PChar) : Integer
6125:  Function StrToIntEx( pszString : PChar; dwFlags : DWORD; var piRet : Integer) : BOOL
6126:  Function StrTrim( psz : PChar; pszTrimChars : PChar) : BOOL
6127:  Function ChrCmpI( w1, w2 : WORD) : BOOL
6128:  Function ChrCmpIA( w1, w2 : WORD) : BOOL
6129:  Function ChrCmpIW( w1, w2 : WORD) : BOOL
6130:  Function StrIntlEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6131:  Function StrIntlEqNI( s1, s2 : PChar; nChar : Integer) : BOOL
6132:  Function StrCatBuff( pszDest, pszSrc : PChar; cchDestBuffSize : Integer) : PChar
6133:  Function StrCpyNX( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6134:  Function IntlStrEqWorker( fCaseSens : BOOL; lpString1, lpString2 : PChar; nChar : Integer) : BOOL
6135:  Function IntlStrEqN( s1, s2 : PChar; nChar : Integer) : BOOL
6136:  SZ_CONTENTTYPE_HTMLA','String').SetString( 'text/html
6137:  SZ_CONTENTTYPE_HTMLW','String').SetString( 'text/html
6138:  SZ_CONTENTTYPE_HTML','string').SetString( SZ_CONTENTTYPE_HTMLA);
6139:  SZ_CONTENTTYPE_CDFA','String').SetString( 'application/x-cdf
6140:  SZ_CONTENTTYPE_CDFW','String').SetString( 'application/x-cdf
6141:  SZ_CONTENTTYPE_CDF','string').SetString( SZ_CONTENTTYPE_CDFA);
6142:  Function PathIsHTMLFile( pszPath : PChar) : BOOL
6143:  STIF_DEFAULT','LongWord').SetUInt( $00000000);
6144:  STIF_SUPPORT_HEX','LongWord').SetUInt( $00000001);
6145:  Function StrNCmpI( lpStr1, lpStr2 : PChar; nChar : Integer) : Integer
6146:  Function StrNCpy( psz1, psz2 : PChar; cchMax : Integer) : PChar
6147:  Function StrCatN( psz1 : PChar; psz2 : PChar; cchMax : Integer) : PChar
6148:  Function PathAddBackslash( pszPath : PChar) : PChar
6149:  Function PathAddExtension( pszPath : PChar; pszExt : PChar) : BOOL
6150:  Function PathAppend( pszPath : PChar; pMore : PChar) : BOOL
6151:  Function PathBuildRoot( szRoot : PChar; iDrive : Integer) : PChar
6152:  Function PathCanonicalize( pszBuf : PChar; pszPath : PChar) : BOOL
6153:  Function PathCombine( szDest : PChar; lpszDir, lpszFile : PChar) : PChar
6154:  Function PathCompactPath( hDC : HDC; pszPath : PChar; dx : UINT) : BOOL
6155:  Function PathCompactPathEx( pszOut : PChar; pszSrc : PChar; cchMax : UINT; dwFlags:DWORD) : BOOL
6156:  Function PathCommonPrefix( pszFile1, pszFile2 : PChar; achPath : PChar) : Integer
6157:  Function PathFileExists( pszPath : PChar) : BOOL
6158:  Function PathFindExtension( pszPath : PChar) : PChar
6159:  Function PathFindFileName( pszPath : PChar) : PChar
6160:  Function PathFindNextComponent( pszPath : PChar) : PChar
```

```
6161:  Function PathFindOnPath( pszPath : PChar; var ppszOtherDirs : PChar) : BOOL
6162:  Function PathGetArgs( pszPath : PChar) : PChar
6163:  Function PathFindSuffixArray(pszPath: PChar; const apszSuffix: PChar; iArraySize: Integer): PChar
6164:  Function PathIsLFNFileSpec( lpName : PChar) : BOOL
6165:  Function PathGetCharType( ch : Char) : UINT
6166:  GCT_INVALID','LongWord').SetUInt( $0000);
6167:  GCT_LFNCHAR','LongWord').SetUInt( $0001);
6168:  GCT_SHORTCHAR','LongWord').SetUInt( $0002);
6169:  GCT_WILD','LongWord').SetUInt( $0004);
6170:  GCT_SEPARATOR','LongWord').SetUInt( $0008);
6171:  Function PathGetDriveNumber( pszPath : PChar) : Integer
6172:  Function PathIsDirectory( pszPath : PChar) : BOOL
6173:  Function PathIsDirectoryEmpty( pszPath : PChar) : BOOL
6174:  Function PathIsFileSpec( pszPath : PChar) : BOOL
6175:  Function PathIsPrefix( pszPrefix, pszPath : PChar) : BOOL
6176:  Function PathIsRelative( pszPath : PChar) : BOOL
6177:  Function PathIsRoot( pszPath : PChar) : BOOL
6178:  Function PathIsSameRoot( pszPath1, pszPath2 : PChar) : BOOL
6179:  Function PathIsUNC( pszPath : PChar) : BOOL
6180:  Function PathIsNetworkPath( pszPath : PChar) : BOOL
6181:  Function PathIsUNCServer( pszPath : PChar) : BOOL
6182:  Function PathIsUNCServerShare( pszPath : PChar) : BOOL
6183:  Function PathIsContentType( pszPath, pszContentType : PChar) : BOOL
6184:  Function PathIsURL( pszPath : PChar) : BOOL
6185:  Function PathMakePretty( pszPath : PChar) : BOOL
6186:  Function PathMatchSpec( pszFile, pszSpec : PChar) : BOOL
6187:  Function PathParseIconLocation( pszIconFile : PChar) : Integer
6188:  Procedure PathQuoteSpaces( lpsz : PChar)
6189:  Function PathRelativePathTo(pszPath:PChar;pszFrom:PChar;dwAttrFrom:DWORD;pszTo:PChar;dwAttrTo:DWORD):BOOL;
6190:  Procedure PathRemoveArgs( pszPath : PChar)
6191:  Function PathRemoveBackslash( pszPath : PChar) : PChar
6192:  Procedure PathRemoveBlanks( pszPath : PChar)
6193:  Procedure PathRemoveExtension( pszPath : PChar)
6194:  Function PathRemoveFileSpec( pszPath : PChar) : BOOL
6195:  Function PathRenameExtension( pszPath : PChar; pszExt : PChar) : BOOL
6196:  Function PathSearchAndQualify( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6197:  Procedure PathSetDlgItemPath( hDlg : HWND; id : Integer; pszPath : PChar)
6198:  Function PathSkipRoot( pszPath : PChar) : PChar
6199:  Procedure PathStripPath( pszPath : PChar)
6200:  Function PathStripToRoot( pszPath : PChar) : BOOL
6201:  Procedure PathUnquoteSpaces( lpsz : PChar)
6202:  Function PathMakeSystemFolder( pszPath : PChar) : BOOL
6203:  Function PathUnmakeSystemFolder( pszPath : PChar) : BOOL
6204:  Function PathIsSystemFolder( pszPath : PChar; dwAttrb : DWORD) : BOOL
6205:  Procedure PathUndecorate( pszPath : PChar)
6206:  Function PathUnExpandEnvStrings( pszPath : PChar; pszBuf : PChar; cchBuf : UINT) : BOOL
6207:  URL_SCHEME_INVALID','LongInt').SetInt( - 1);
6208:  URL_SCHEME_UNKNOWN','LongInt').SetInt( 0);
6209:  URL_SCHEME_FTP','LongInt').SetInt( 1);
6210:  URL_SCHEME_HTTP','LongInt').SetInt( 2);
6211:  URL_SCHEME_GOPHER','LongInt').SetInt( 3);
6212:  URL_SCHEME_MAILTO','LongInt').SetInt( 4);
6213:  URL_SCHEME_NEWS','LongInt').SetInt( 5);
6214:  URL_SCHEME_NNTP','LongInt').SetInt( 6);
6215:  URL_SCHEME_TELNET','LongInt').SetInt( 7);
6216:  URL_SCHEME_WAIS','LongInt').SetInt( 8);
6217:  URL_SCHEME_FILE','LongInt').SetInt( 9);
6218:  URL_SCHEME_MK','LongInt').SetInt( 10);
6219:  URL_SCHEME_HTTPS','LongInt').SetInt( 11);
6220:  URL_SCHEME_SHELL','LongInt').SetInt( 12);
6221:  URL_SCHEME_SNEWS','LongInt').SetInt( 13);
6222:  URL_SCHEME_LOCAL','LongInt').SetInt( 14);
6223:  URL_SCHEME_JAVASCRIPT','LongInt').SetInt( 15);
6224:  URL_SCHEME_VBSCRIPT','LongInt').SetInt( 16);
6225:  URL_SCHEME_ABOUT','LongInt').SetInt( 17);
6226:  URL_SCHEME_RES','LongInt').SetInt( 18);
6227:  URL_SCHEME_MAXVALUE','LongInt').SetInt( 19);
6228:  URL_SCHEME', 'Integer
6229:  URL_PART_NONE','LongInt').SetInt( 0);
6230:  URL_PART_SCHEME','LongInt').SetInt( 1);
6231:  URL_PART_HOSTNAME','LongInt').SetInt( 2);
6232:  URL_PART_USERNAME','LongInt').SetInt( 3);
6233:  URL_PART_PASSWORD','LongInt').SetInt( 4);
6234:  URL_PART_PORT','LongInt').SetInt( 5);
6235:  URL_PART_QUERY','LongInt').SetInt( 6);
6236:  URL_PART', 'DWORD
6237:  URLIS_URL','LongInt').SetInt( 0);
6238:  URLIS_OPAQUE','LongInt').SetInt( 1);
6239:  URLIS_NOHISTORY','LongInt').SetInt( 2);
6240:  URLIS_FILEURL','LongInt').SetInt( 3);
6241:  URLIS_APPLIABLE','LongInt').SetInt( 4);
6242:  URLIS_DIRECTORY','LongInt').SetInt( 5);
6243:  URLIS_HASQUERY','LongInt').SetInt( 6);
6244:  TUrlIs', 'DWORD
6245:  URL_UNESCAPE','LongWord').SetUInt( $10000000);
6246:  URL_ESCAPE_UNSAFE','LongWord').SetUInt( $20000000);
6247:  URL_PLUGGABLE_PROTOCOL','LongWord').SetUInt( $40000000);
6248:  URL_WININET_COMPATIBILITY','LongWord').SetUInt( DWORD ( $80000000 ));
6249:  URL_DONT_ESCAPE_EXTRA_INFO','LongWord').SetUInt( $02000000);
```

```
6250:  URL_ESCAPE_SPACES_ONLY','LongWord').SetUInt( $04000000);
6251:  URL_DONT_SIMPLIFY','LongWord').SetUInt( $08000000);
6252:  URL_NO_META','longword').SetUInt( URL_DONT_SIMPLIFY);
6253:  URL_UNESCAPE_INPLACE','LongWord').SetUInt( $00100000);
6254:  URL_CONVERT_IF_DOSPATH','LongWord').SetUInt( $00200000);
6255:  URL_UNESCAPE_HIGH_ANSI_ONLY','LongWord').SetUInt( $00400000);
6256:  URL_INTERNAL_PATH','LongWord').SetUInt( $00800000);
6257:  URL_FILE_USE_PATHURL','LongWord').SetUInt( $00010000);
6258:  URL_ESCAPE_PERCENT','LongWord').SetUInt( $00001000);
6259:  URL_ESCAPE_SEGMENT_ONLY','LongWord').SetUInt( $00002000);
6260:  URL_PARTFLAG_KEEPSCHEME','LongWord').SetUInt( $00000001);
6261:  URL_APPLY_DEFAULT','LongWord').SetUInt( $00000001);
6262:  URL_APPLY_GUESSSCHEME','LongWord').SetUInt( $00000002);
6263:  URL_APPLY_GUESSFILE','LongWord').SetUInt( $00000004);
6264:  URL_APPLY_FORCEAPPLY','LongWord').SetUInt( $00000008);
6265:  Function UrlCompare( psz1, psz2 : PChar; fIgnoreSlash : BOOL) : Integer
6266:  Function UrlCombine(pszBase,pszRelative:PChar;pszCombin:PChar;out pcchCombin:DWORD;dwFlags:DWORD):HRESULT;
6267:  Function UrlCanonicalize(pszUrl:PChar;pszCanonicalized:PChar;pcchCanonic:DWORD;dwFlags:DWORD):HRESULT;
6268:  Function UrlIsOpaque( pszURL : PChar) : BOOL
6269:  Function UrlIsNoHistory( pszURL : PChar) : BOOL
6270:  Function UrlIsFileUrl( pszURL : PChar) : BOOL
6271:  Function UrlIs( pszUrl : PChar; UrlIs : TUrlIs) : BOOL
6272:  Function UrlGetLocation( psz1 : PChar) : PChar
6273:  Function UrlUnescape( pszUrl, pszUnescaped : PChar;pcchUnescaped:DWORD; dwFlags : DWORD) : HRESULT
6274:  Function UrlEscape(pszUrl: PChar; pszEscaped: PChar; pcchEscaped:DWORD; dwFlags : DWORD) : HRESULT
6275:  Function UrlCreateFromPath(pszPath:PChar; pszUrl: PChar;pcchUrl: DWORD; dwFlags : DWORD) : HRESULT
6276:  Function PathCreateFromUrl(pszUrl:PChar; pszPath:PChar; pcchPath:DWORD; dwFlags : DWORD) : HRESULT
6277:  Function UrlHash( pszUrl : PChar; pbHash : BYTE; cbHash : DWORD) : HRESULT
6278:  Function UrlGetPart(pszIn: PChar; pszOut: PChar; pcchOut: DWORD; dwPart,dwFlags: DWORD) : HRESULT
6279:  Function UrlApplyScheme( pszIn : PChar; pszOut : PChar; pcchOut : DWORD; dwFlags : DWORD) : HRESULT
6280:  Function HashData( pbData : BYTE; cbData : DWORD; pbHash : BYTE; cbHash : DWORD) : HRESULT
6281:  Function UrlEscapeSpaces( pszUrl : PChar; pszEscaped : PChar; pcchEscaped : DWORD) : HRESULT
6282:  Function UrlUnescapeInPlace( pszUrl : PChar; dwFlags : DWORD) : HRESULT
6283:  Function SHDeleteEmptyKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6284:  Function SHDeleteKey( hKey : HKEY; pszSubKey : PChar) : DWORD
6285:  Function SHDeleteValue( hKey : HKEY; pszSubKey, pszValue : PChar) : DWORD
6286:  Function SHEnumKeyEx( hKey : HKEY; dwIndex : DWORD; pszName : PChar; var pcchName : DWORD) : Longint
6287:  Function SHEnumValue( hKey : HKEY; dwIndex : DWORD; pszValueName:PChar; var
      pcchValueName:DWORD;pdwType:DWORD; pvData : ___Pointer; pcbData : DWORD) : Longint
6288:  Function SHQueryInfoKey(hKey:HKEY;pcSubKeys,pcchMaxSubKeyLen,pcVal,pcchMaxValueNameLen:DWORD):Longint;
6289:  Function SHCopyKey( hkeySrc : HKEY; szSrcSubKey : PChar; hkeyDest : HKEY; fReserved : DWORD) : DWORD
6290:  Function SHRegGetPath(hKey:HKEY; pcszSubKey,pcszValue: PChar; pszPath: PChar; dwFlags: DWORD): DWORD
6291:  Function SHRegSetPath( hKey:HKEY; pcszSubKey, pcszValue, pcszPath : PChar; dwFlags : DWORD): DWORD
6292:  SHREGDEL_DEFAULT','LongWord').SetUInt( $00000000);
6293:  SHREGDEL_HKCU','LongWord').SetUInt( $00000001);
6294:  SHREGDEL_HKLM','LongWord').SetUInt( $00000010);
6295:  SHREGDEL_BOTH','LongWord').SetUInt( $00000011);
6296:  SHREGENUM_DEFAULT','LongWord').SetUInt( $00000000);
6297:  SHREGENUM_HKCU','LongWord').SetUInt( $00000001);
6298:  SHREGENUM_HKLM','LongWord').SetUInt( $00000010);
6299:  SHREGENUM_BOTH','LongWord').SetUInt( $00000011);
6300:  SHREGSET_HKCU','LongWord').SetUInt( $00000001);
6301:  SHREGSET_FORCE_HKCU','LongWord').SetUInt( $00000002);
6302:  SHREGSET_HKLM','LongWord').SetUInt( $00000004);
6303:  SHREGSET_FORCE_HKLM','LongWord').SetUInt( $00000008);
6304:  TSHRegDelFlags', 'DWORD
6305:  TSHRegEnumFlags', 'DWORD
6306:  HUSKEY', 'THandle
6307:  ASSOCF_INIT_NOREMAPCLSID','LongWord').SetUInt( $00000001);
6308:  ASSOCF_INIT_BYEXENAME','LongWord').SetUInt( $00000002);
6309:  ASSOCF_OPEN_BYEXENAME','LongWord').SetUInt( $00000002);
6310:  ASSOCF_INIT_DEFAULTTOSTAR','LongWord').SetUInt( $00000004);
6311:  ASSOCF_INIT_DEFAULTTOFOLDER','LongWord').SetUInt( $00000008);
6312:  ASSOCF_NOUSERSETTINGS','LongWord').SetUInt( $00000010);
6313:  ASSOCF_NOTRUNCATE','LongWord').SetUInt( $00000020);
6314:  ASSOCF_VERIFY','LongWord').SetUInt( $00000040);
6315:  ASSOCF_REMAPRUNDLL','LongWord').SetUInt( $00000080);
6316:  ASSOCF_NOFIXUPS','LongWord').SetUInt( $00000100);
6317:  ASSOCF_IGNOREBASECLASS','LongWord').SetUInt( $00000200);
6318:  ASSOCF', 'DWORD
6319:  ASSOCSTR_COMMAND','LongInt').SetInt( 1);
6320:  ASSOCSTR_EXECUTABLE','LongInt').SetInt( 2);
6321:  ASSOCSTR_FRIENDLYDOCNAME','LongInt').SetInt( 3);
6322:  ASSOCSTR_FRIENDLYAPPNAME','LongInt').SetInt( 4);
6323:  ASSOCSTR_NOOPEN','LongInt').SetInt( 5);
6324:  ASSOCSTR_SHELLNEWVALUE','LongInt').SetInt( 6);
6325:  ASSOCSTR_DDECOMMAND','LongInt').SetInt( 7);
6326:  ASSOCSTR_DDEIFEXEC','LongInt').SetInt( 8);
6327:  ASSOCSTR_DDEAPPLICATION','LongInt').SetInt( 9);
6328:  ASSOCSTR_DDETOPIC','LongInt').SetInt( 10);
6329:  ASSOCSTR_INFOTIP','LongInt').SetInt( 11);
6330:  ASSOCSTR_MAX','LongInt').SetInt( 12);
6331:  ASSOCSTR', 'DWORD
6332:  ASSOCKEY_SHELLEXECCLASS','LongInt').SetInt( 1);
6333:  ASSOCKEY_APP','LongInt').SetInt( 2);
6334:  ASSOCKEY_CLASS','LongInt').SetInt( 3);
6335:  ASSOCKEY_BASECLASS','LongInt').SetInt( 4);
6336:  ASSOCKEY_MAX','LongInt').SetInt( 5);
6337:  ASSOCKEY', 'DWORD
```

```
6338:  ASSOCDATA_MSIDESCRIPTOR','LongInt').SetInt( 1);
6339:  ASSOCDATA_NOACTIVATEHANDLER','LongInt').SetInt( 2);
6340:  ASSOCDATA_QUERYCLASSSTORE','LongInt').SetInt( 3);
6341:  ASSOCDATA_HASPERUSERASSOC','LongInt').SetInt( 4);
6342:  ASSOCDATA_MAX','LongInt').SetInt( 5);
6343:  ASSOCDATA', 'DWORD
6344:  ASSOCENUM_NONE','LongInt').SetInt( 0);
6345:  ASSOCENUM', 'DWORD
6346:  SID_IQueryAssociations','String').SetString( '{c46ca590-3c3f-11d2-bee6-0000f805ca57}
6347:  SHACF_DEFAULT','LongWord').SetUInt( $00000000);
6348:  SHACF_FILESYSTEM','LongWord').SetUInt( $00000001);
6349:  SHACF_URLHISTORY','LongWord').SetUInt( $00000002);
6350:  SHACF_URLMRU','LongWord').SetUInt( $00000004);
6351:  SHACF_USETAB','LongWord').SetUInt( $00000008);
6352:  SHACF_FILESYS_ONLY','LongWord').SetUInt( $00000010);
6353:  SHACF_AUTOSUGGEST_FORCE_ON','LongWord').SetUInt( $10000000);
6354:  SHACF_AUTOSUGGEST_FORCE_OFF','LongWord').SetUInt( $20000000);
6355:  SHACF_AUTOAPPEND_FORCE_ON','LongWord').SetUInt( $40000000);
6356:  SHACF_AUTOAPPEND_FORCE_OFF','LongWord').SetUInt( DWORD ( $80000000 ));
6357:  Function SHAutoComplete( hwndEdit : HWND; dwFlags : DWORD) : HRESULT
6358:  Procedure SHSetThreadRef( punk : IUnknown)
6359:  Procedure SHGetThreadRef( out ppunk : IUnknown)
6360:  CTF_INSIST','LongWord').SetUInt( $00000001);
6361:  CTF_THREAD_REF','LongWord').SetUInt( $00000002);
6362:  CTF_PROCESS_REF','LongWord').SetUInt( $00000004);
6363:  CTF_COINIT','LongWord').SetUInt( $00000008);
6364:  Function SHCreateShellPalette( hdc : HDC) : HPALETTE
6365:  Procedure ColorRGBToHLS( clrRGB : TColorRef; out pwHue, pwLuminance, pwSaturation : WORD)
6366:  Function ColorHLSToRGB( wHue, wLuminance, wSaturation : WORD) : TColorRef
6367:  Function ColorAdjustLuma( clrRGB : TColorRef; n : Integer; fScale : Boolean) : TColorRef
6368:  Function GetSysColorBrush( nIndex : Integer) : HBRUSH
6369:  Function DrawFocusRect( hDC : HDC; const lprc : TRect) : BOOL
6370:  Function FillRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6371:  Function FrameRect( hDC : HDC; const lprc : TRect; hbr : HBRUSH) : Integer
6372:  Function InvertRect( hDC : HDC; const lprc : TRect) : BOOL
6373:  Function SetRect( var lprc : TRect; xLeft, yTop, xRight, yBottom : Integer) : BOOL
6374:  Function SetRectEmpty( var lprc : TRect) : BOOL
6375:  Function CopyRect( var lprcDst : TRect; const lprcSrc : TRect) : BOOL
6376:  Function InflateRect( var lprc : TRect; dx, dy : Integer) : BOOL
6377:  Function IntersectRect2( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6378:  Function SubtractRect( var lprcDst : TRect; const lprcSrc1, lprcSrc2 : TRect) : BOOL
6379:
6380:  Function InitializeFlatSB( hWnd : HWND) : Bool
6381:  Procedure UninitializeFlatSB( hWnd : HWND)
6382:  Function FlatSB_GetScrollProp( p1 : HWND; propIndex : Integer; p3 : PInteger) : Bool
6383:  Function FlatSB_SetScrollProp( p1 : HWND; index : Integer; newValue : Integer; p4 : Bool) : Bool
6384:  Function GET_APPCOMMAND_LPARAM( lParam : Integer) : Word  //of JvWin32
6385:  Function GET_DEVICE_LPARAM( lParam : Integer) : Word
6386:  Function GET_MOUSEORKEY_LPARAM( lParam : Integer) : Integer
6387:  Function GET_FLAGS_LPARAM( lParam : Integer) : Word
6388:  Function GET_KEYSTATE_LPARAM( lParam : Integer) : Word
6389:
6390:
6391: //  ****************************************** 204 unit uPSI_ShellAPI;
6392: Function DragQueryFile( Drop : HDROP; FileIndex : UINT; FileName : PChar; cb : UINT) : UINT
6393: Function DragQueryPoint( Drop : HDROP; var Point : TPoint) : BOOL
6394: Procedure DragFinish( Drop : HDROP)
6395: Procedure DragAcceptFiles( Wnd : HWND; Accept : BOOL)
6396: Function ShellExecute(hWnd:HWND;Operation,FileName,Parameters,Directory:PChar;ShowCmd:Integer):HINST
6397: Function FindExecutable( FileName, Directory : PChar; Result : PChar) : HINST
6398: Function ShellAbout( Wnd : HWND; szApp, szOtherStuff : PChar; Icon : HICON) : Integer
6399: Function DuplicateIcon( hInst : HINST; Icon : HICON) : HICON
6400: Function ExtractAssociatedIcon( hInst : HINST; lpIconPath : PChar; var lpiIcon : Word) : HICON
6401: Function ExtractIcon( hInst : HINST; lpszExeFileName : PChar; nIconIndex : UINT) : HICON
6402: Function SHAppBarMessage( dwMessage : DWORD; var pData : TAppBarData) : UINT
6403: Function DoEnvironmentSubst( szString : PChar; cbString : UINT) : DWORD
6404: Function ExtractIconEx(lpszFile:PChar;nIconIndex:Int;var phiconLarge,phiconSmall:HICON;nIcons:UINT):UINT;
6405: Procedure SHFreeNameMappings( hNameMappings : THandle)
6406:
6407:  DLLVER_PLATFORM_WINDOWS','LongWord').SetUInt( $00000001);
6408:  DLLVER_PLATFORM_NT','LongWord').SetUInt( $00000002);
6409:  DLLVER_MAJOR_MASK','LongWord').SetUInt( Int64 ( $FFFF000000000000 ));
6410:  DLLVER_MINOR_MASK','LongWord').SetUInt( Int64 ( $0000FFFF00000000 ));
6411:  DLLVER_BUILD_MASK','LongWord').SetUInt( Int64 ( $00000000FFFF0000 ));
6412:  DLLVER_QFE_MASK','LongWord').SetUInt( Int64 ( $000000000000FFFF ));
6413:  Function MAKEDLLVERULL( Major, Minor, Build, Qfe : Word) : Int64
6414:  Function SimpleXMLEncode( const S : string) : string
6415:  Procedure SimpleXMLDecode( var S : string; TrimBlanks : Boolean)
6416:  Function XMLEncode( const S : string) : string
6417:  Function XMLDecode( const S : string) : string
6418:  Function EntityEncode( const S : string) : string
6419:  Function EntityDecode( const S : string) : string
6420:
6421: procedure RIRegister_CPort_Routines(S: TPSExec);
6422:  Procedure EnumComPorts( Ports : TStrings)
6423:  Procedure ListComPorts( Ports : TStrings)
6424:  Procedure ComPorts( Ports : TStrings)  //Alias to Arduino
6425:  Function GetComPorts: TStringlist;
6426:  Function StrToBaudRate( Str : string) : TBaudRate
```

```
6427:  Function StrToStopBits( Str : string) : TStopBits
6428:  Function StrToDataBits( Str : string) : TDataBits
6429:  Function StrToParity( Str : string) : TParityBits
6430:  Function StrToFlowControl( Str : string) : TFlowControl
6431:  Function BaudRateToStr( BaudRate : TBaudRate) : string
6432:  Function StopBitsToStr( StopBits : TStopBits) : string
6433:  Function DataBitsToStr( DataBits : TDataBits) : string
6434:  Function ParityToStr( Parity : TParityBits) : string
6435:  Function FlowControlToStr( FlowControl : TFlowControl) : string
6436:  Function ComErrorsToStr( Errors : TComErrors) : String
6437:
6438:  Function GetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
6439:  Function DispatchMessage( const lpMsg : TMsg) : Longint
6440:  Function TranslateMessage( const lpMsg : TMsg) : BOOL
6441:  Function SetMessageQueue( cMessagesMax : Integer) : BOOL
6442:  Function PeekMessage(var lpMsg:TMsg; hWnd:HWND;wMsgFilterMin,wMsgFilterMax,wRemoveMsg:UINT):BOOL
6443:  Function GetMessagePos : DWORD
6444:  Function GetMessageTime : Longint
6445:  Function GetMessageExtraInfo : Longint
6446:  Function GetSpecialFolderPath( const FolderName : string; CanCreate : Boolean) : string
6447:  Procedure JAddToRecentDocs( const Filename : string)
6448:  Procedure ClearRecentDocs
6449:  Function ExtractIconFromFile( FileName : string; Index : Integer) : HICON
6450:  Function CreateShellLink( const AppName, Desc : string; Dest : string) : string
6451:  Procedure GetShellLinkInfo( const LinkFile : WideString; var SLI : TShellLinkInfo)
6452:  Procedure SetShellLinkInfo( const LinkFile : WideString; const SLI : TShellLinkInfo)
6453:  Function RecycleFile( FileToRecycle : string) : Boolean
6454:  Function JCopyFile( FromFile, ToDir : string) : Boolean
6455:  Function ShellObjectTypeEnumToConst( ShellObjectType : TShellObjectType) : UINT
6456:  Function ShellObjectTypeConstToEnum( ShellObjectType : UINT) : TShellObjectType
6457:  Function QueryServiceConfig2A( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6458:  Function QueryServiceConfig2W( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6459:  Function QueryServiceConfig2( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
      DWORD; var pcbBytesNeeded : DWORD) : BOOL
6460:  Function EnumServicesStatusExA(hSCManager: SC_HANDLE; InfoLevel: SC_ENUM_TYPE; dwServiceType: DWORD;
      dwServiceState: DWORD;lpServices:LPBYTE;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD;pszGroupName: LP
6461:  Function EnumServicesStatusExW( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
      dwServiceState : DWORD; lpServices:LPBYTE; cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD; pszGroupNam
6462:  Function EnumServicesStatusEx( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
      dwServiceState : DWORD;lpServices :LPBYTE;cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
      lpResumeHandle:DWORD; pszGroupName
6463:  Function ConvertSidToStringSid( sid : PSID; var stringSid : LPWSTR) : BOOL
6464:
6465: ****************************************** unit uPSI_JclPeImage;
6466:
6467:  Function IsValidPeFile( const FileName : TFileName) : Boolean
6468: // Function PeGetNtHeaders( const FileName : TFileName; var NtHeaders : TImageNtHeaders) : Boolean
6469:  Function PeCreateNameHintTable( const FileName : TFileName) : Boolean
6470:  Function PeRebaseImage(const ImageName: TFileName; NewBase : DWORD; TimeStamp : DWORD; MaxNewSize :
      DWORD) : TJclRebaseImageInfo
6471:  Function PeVerifyCheckSum( const FileName : TFileName) : Boolean
6472:  Function PeClearCheckSum( const FileName : TFileName) : Boolean
6473:  Function PeUpdateCheckSum( const FileName : TFileName) : Boolean
6474:  Function PeDoesExportFunction(const FileName:TFileName;const
      FuncName:string;Options:TJclSmartCompOptions):Bool;
6475:  Function PeIsExportFunctionForwardedEx( const FileName : TFileName; const FunctionName : string; var
      ForwardedName : string; Options : TJclSmartCompOptions) : Boolean
6476:  Function PeIsExportFunctionForwarded( const FileName : TFileName; const FunctionName : string; Options :
      TJclSmartCompOptions) : Boolean
6477:  Function PeDoesImportFunction( const FileName : TFileName; const FunctionName : string; const LibraryName
      : string; Options : TJclSmartCompOptions) : Boolean
6478:  Function PeDoesImportLibrary(const FileName:TFileName;const
      LibraryName:string;Recursive:Boolean):Boolean);
6479:  Function PeImportedLibraries( const FileName : TFileName; const LibrariesList : TStrings; Recursive :
      Boolean; FullPathName : Boolean) : Boolean
6480:  Function PeImportedFunctions( const FileName : TFileName; const FunctionsList : TStrings; const
      LibraryName : string; IncludeLibNames : Boolean) : Boolean
6481:  Function PeExportedFunctions( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6482:  Function PeExportedNames( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6483:  Function PeExportedVariables( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6484:  Function PeResourceKindNames(const FileN:TFileName;ResourceType:TJclPeResourceKind;const
      NamesList:TStrings):Boolean;
6485:  Function PeBorFormNames( const FileName : TFileName; const NamesList : TStrings) : Boolean
6486:  Function PeBorDependedPackages(const FileName:TFileName;PackagesList:TStrings;FullPathName,
      Descript:Bool):Bool;
6487:  Function PeFindMissingImports( const FileName : TFileName; MissingImportsList : TStrings) : Boolean;
6488:  Function PeFindMissingImports1( RequiredImportsList, MissingImportsList : TStrings) : Boolean;
6489:  Function PeCreateRequiredImportList(const FileName: TFileName; RequiredImportsList: TStrings): Boolean;
6490:  //Function PeMapImgNtHeaders( const BaseAddress : Pointer) : PImageNtHeaders
6491:  //Function PeMapImgLibraryName( const BaseAddress : Pointer) : string
6492:  //Function PeMapImgSections( const NtHeaders : PImageNtHeaders) : PImageSectionHeader
6493:  //Function PeMapImgFindSection( const NtHeaders : PImageNtHeaders; const SectionName : string) :
      PImageSectionHeader
6494:  //Function PeMapImgExportedVariables(const Module: HMODULE; const VariablesList:TStrings):Boolean
6495:  //Function PeMapImgResolvePackageThunk( Address : Pointer) : Pointer
```

```
6496:  Function PeMapFindResource(const Module:HMODULE;const ResourceType:PChar;const ResourceName:string):
       ___Pointer;
6497:   SIRegister_TJclPeSectionStream(CL);
6498:   SIRegister_TJclPeMapImgHookItem(CL);
6499:   SIRegister_TJclPeMapImgHooks(CL);
6500:  //Function PeDbgImgNtHeaders(ProcessHandle:THandle;BaseAddress:Pointer;var
       NtHeaders:TImageNtHeaders):Boolean
6501:  //Function PeDbgImgLibraryName(ProcessHandle:THandle; BaseAddress:Pointer; var Name:string):Boolean
6502:   Type TJclBorUmSymbolKind','(skData,skFunction,skConstructor,skDestructor,skRTTI,skVTable)
6503:   TJclBorUmSymbolModifier', '( smQualified, smLinkProc )
6504:   TJclBorUmSymbolModifiers', 'set of TJclBorUmSymbolModifier
6505:   TJclBorUmDescription', 'record Kind : TJclBorUmSymbolKind; Modifiers : TJclBorUmSymbolModifiers; end
6506:   TJclBorUmResult', '( urOk, urNotMangled, urMicrosoft, urError )
6507:   TJclPeUmResult', '( umNotMangled, umBorland, umMicrosoft )
6508:  Function PeBorUnmangleName( const Name : string; var Unmangled : string; var Description :
       TJclBorUmDescription; var BasePos : Integer) : TJclBorUmResult;
6509:  Function PeBorUnmangleName1(const Name:string;var Unmangled:string;var
       Descript:TJclBorUmDescription):TJclBorUmResult;
6510:  Function PeBorUnmangleName2( const Name : string; var Unmangled : string) : TJclBorUmResult;
6511:  Function PeBorUnmangleName3( const Name : string) : string;
6512:  Function PeIsNameMangled( const Name : string) : TJclPeUmResult
6513:  Function PeUnmangleName( const Name : string; var Unmangled : string) : TJclPeUmResult
6514:
6515:
6516: //***************** SysTools uPSI_StSystem; ************************************
6517:  Function StCopyFile( const SrcPath, DestPath : AnsiString) : Cardinal
6518:  Function CreateTempFile( const aFolder : AnsiString; const aPrefix : AnsiString) : AnsiString
6519:  Function DeleteVolumeLabel( Drive : Char) : Cardinal
6520:  //Procedure EnumerateDirectories(const
       StartDir:AnsiString;FL:TStrings;SubDirs:Bool;IncludeItem:TIncludeItemFunc);
6521:  //Procedure EnumerateFiles(const StartDir:AnsiString;FL:TStrings;SubDirs:Bool
       IncludeItem:TIncludeItemFunc);
6522:  Function FileHandlesLeft( MaxHandles : Cardinal) : Cardinal
6523:  Function FileMatchesMask( const FileName, FileMask : AnsiString) : Boolean
6524:  Function FileTimeToStDateTime( FileTime : LongInt) : TStDateTimeRec
6525:  Function FindNthSlash( const Path : AnsiString; n : Integer) : Integer
6526:  Function FlushOsBuffers( Handle : Integer) : Boolean
6527:  Function GetCurrentUser : AnsiString
6528:  Function GetDiskClass( Drive : Char) : DiskClass
6529:  Function GetDiskInfo(Drive:Char;var ClustersAvail,TotalClusters,BytesPerSector,
       SectorsPerCluster:Cardinal):Bool;
6530:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Double;var TotalSpaceAvail:Double; var
       DiskSize:Double):Boolean;
6531:  Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Comp;var TotalSpaceAvail:Comp;var
       DiskSize:Comp):Boolean;
6532:   { index 0 - FreeBytesAvailable, 1 - TotalNumberOfBytes, 2 - TotalNumberOfFreeBytes  }
6533:  Function getDiskSpace2(const path: String; index: integer): int64;
6534:  Function GetFileCreateDate( const FileName : AnsiString) : TDateTime
6535:  Function StGetFileLastAccess( const FileName : AnsiString) : TDateTime
6536:  Function GetFileLastModify( const FileName : AnsiString) : TDateTime
6537:  Function GetHomeFolder( aForceSlash : Boolean) : AnsiString
6538:  Function GetLongPath( const APath : AnsiString) : AnsiString
6539:  Function GetMachineName : AnsiString
6540:  Function GetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6541:  Function GetParentFolder( const APath : AnsiString; aForceSlash : Boolean) : AnsiString
6542:  Function GetShortPath( const APath : AnsiString) : AnsiString
6543:  Function GetSystemFolder( aForceSlash : Boolean) : AnsiString
6544:  Function GetTempFolder( aForceSlash : boolean) : AnsiString
6545:  Function StGetWindowsFolder( aForceSlash : boolean) : AnsiString
6546:  Function GetWorkingFolder( aForceSlash : boolean) : AnsiString
6547:  Function GlobalDateTimeToLocal( const UTC : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6548:  Function StIsDirectory( const DirName : AnsiString) : Boolean
6549:  Function IsDirectoryEmpty( const S : AnsiString) : Integer
6550:  Function IsDriveReady( Drive : Char) : Boolean
6551:  Function IsFile( const FileName : AnsiString) : Boolean
6552:  Function IsFileArchive( const S : AnsiString) : Integer
6553:  Function IsFileHidden( const S : AnsiString) : Integer
6554:  Function IsFileReadOnly( const S : AnsiString) : Integer
6555:  Function IsFileSystem( const S : AnsiString) : Integer
6556:  Function LocalDateTimeToGlobal( const DT1 : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6557:  Function ReadVolumeLabel( var VolName : AnsiString; Drive : Char) : Cardinal
6558:  Function SameFile( const FilePath1, FilePath2 : AnsiString; var ErrorCode : Integer) : Boolean
6559:  Function SetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6560:  Procedure SplitPath( const APath : AnsiString; Parts : TStrings)
6561:  Function StDateTimeToFileTime( const FileTime : TStDateTimeRec) : LongInt
6562:  Function StDateTimeToUnixTime( const DT1 : TStDateTimeRec) : Longint
6563:  Function UnixTimeToStDateTime( UnixTime : Longint) : TStDateTimeRec
6564:  Function ValidDrive( Drive : Char) : Boolean
6565:  Function WriteVolumeLabel( const VolName : AnsiString; Drive : Char) : Cardinal
6566:
6567: //****************************unit uPSI_JclLANMan;*************************************
6568: Function CreateAccount( const Server, Username, Fullname, Password, Description, Homedir, Script : string;
       const PasswordNeverExpires : Boolean) : Boolean
6569:  Function CreateLocalAccount( const Username, Fullname, Password, Description, Homedir, Script : string;
       const PasswordNeverExpires : Boolean) : Boolean
6570:  Function DeleteAccount( const Servername, Username : string) : Boolean
6571:  Function DeleteLocalAccount( Username : string) : Boolean
6572:  Function CreateLocalGroup( const Server, Groupname, Description : string) : Boolean
6573:  Function CreateGlobalGroup( const Server, Groupname, Description : string) : Boolean
```

```
6574:  Function DeleteLocalGroup( const Server, Groupname : string) : Boolean
6575:  Function GetLocalGroups( const Server : string; const Groups : TStrings) : Boolean
6576:  Function GetGlobalGroups( const Server : string; const Groups : TStrings) : Boolean
6577:  Function LocalGroupExists( const Group : string) : Boolean
6578:  Function GlobalGroupExists( const Server, Group : string) : Boolean
6579:  Function AddAccountToLocalGroup( const Accountname, Groupname : string) : Boolean
6580:  Function LookupGroupName( const Server : string; const RID : TNetWellKnownRID) : string
6581:  Procedure ParseAccountName( const QualifiedName : string; var Domain, UserName : string)
6582:  Function IsLocalAccount( const AccountName : string) : Boolean
6583:  Function TimeStampInterval( StartStamp, EndStamp : TDateTime) : integer
6584:  Function GetRandomString( NumChar : cardinal) : string
6585:
6586: //****************************unit uPSI_cUtils;*****************************************
6587: TypeS('TUnitType', '( utSrc, utHead, utRes, utPrj, utOther )
6588: Function cIsWinNT : boolean
6589:  Procedure cFilesFromWildcard(Directory,Mask: string;var Files:TStringList;Subdirs,ShowDirs,
      Multitasking:Boolean;
6590:  Function cExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
6591:  Function cRunAndGetOutput(Cmd,WorkDir:string; ErrFunc:TErrFunc; LineOutputFunc:TLineOutputFunc;
      CheckAbortFunc : TCheckAbortFunc; ShowReturnValue : Boolean) : string
6592:  Function cGetShortName( FileName : string) : string
6593:  Procedure cShowError( Msg : String)
6594:  Function cCommaStrToStr( s : string; formatstr : string) : string
6595:  Function cIncludeQuoteIfSpaces( s : string) : string
6596:  Function cIncludeQuoteIfNeeded( s : string) : string
6597:  Procedure cLoadFilefromResource( const FileName : string; ms : TMemoryStream)
6598:  Function cValidateFile(const FileName:string; const WorkPath:string;const CheckDirs:boolean):string;
6599:  Function cBuildFilter( var value : string; const FLTStyle : TFILTERSET) : boolean;
6600:  Function cBuildFilter1( var value : string; const _filters : array of string) : boolean;
6601:  Function cCodeInstoStr( s : string) : string
6602:  Function cStrtoCodeIns( s : string) : string
6603:  Procedure cStrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6604:  Function cAttrtoStr( const Attr : TSynHighlighterAttributes) : string
6605:  Procedure cStrtoPoint( var pt : TPoint; value : string)
6606:  Function cPointtoStr( const pt : TPoint) : string
6607:  Function cListtoStr( const List : TStrings) : string
6608:  Function ListtoStr( const List : TStrings) : string
6609:  Procedure StrtoList( s : string; const List : TStrings; const delimiter : char)
6610:  Procedure cStrtoList( s : string; const List : TStrings; const delimiter : char)
6611:  Function cGetFileTyp( const FileName : string) : TUnitType
6612:  Function cGetExTyp( const FileName : string) : TExUnitType
6613:  Procedure cSetPath( Add : string; const UseOriginal : boolean)
6614:  Function cExpandFileto( const FileName : string; const BasePath : string) : string
6615:  Function cFileSamePath( const FileName : string; const TestPath : string) : boolean
6616:  Procedure cCloneMenu( const FromMenu : TMenuItem; ToMenu : TMenuItem)
6617:  Function cGetLastPos( const SubStr : string; const S : string) : integer
6618:  Function cGenMakePath( FileName : String) : String;
6619:  Function cGenMakePath2( FileName : String) : String
6620:  Function cGenMakePath1( FileName : String; EscapeSpaces, EncloseInQuotes : Boolean) : String;
6621:  Function cGetRealPath( BrokenFileName : String; Directory : String) : String
6622:  Function cCalcMod( Count : Integer) : Integer
6623:  Function cGetVersionString( FileName : string) : string
6624:  Function cCheckChangeDir( var Dir : string) : boolean
6625:  Function cGetAssociatedProgram(const Extension:string; var Filename,Description: string):boolean
6626:  Function cIsNumeric( s : string) : boolean
6627:  Procedure StrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6628:  Function AttrtoStr( const Attr : TSynHighlighterAttributes) : string
6629:  Function GetFileTyp( const FileName : string) : TUnitType
6630:  Function Atoi(const aStr: string): integer
6631:  Function Itoa(const aint: integer): string
6632:
6633:
6634: procedure SIRegister_cHTTPUtils(CL: TPSPascalCompiler);
6635: begin
6636:   FindClass('TOBJECT'),'EHTTP
6637:   FindClass('TOBJECT'),'EHTTPParser
6638:   //AnsiCharSet', 'set of AnsiChar
6639:   AnsiStringArray', 'array of AnsiString
6640:   THTTPProtocolEnum', '( hpNone, hpCustom, hpHTTP, hpHTTPS )
6641:   THTTPVersionEnum', '( hvNone, hvCustom, hvHTTP10, hvHTTP11 )
6642:   THTTPVersion', 'record Version : THTTPVersionEnum; Protocol : TH'
6643:    +'TTPProtocolEnum; CustomProtocol : AnsiString; CustomMajVersion : Integer; '
6644:    +'CustomMinVersion : Integer; end
6645:   THTTPHeaderNameEnum', '( hntCustom, hntHost, hntContentType, hnt'
6646:    +'ContentLength, hntContentTransferEncoding, hntContentLocation, hntContentL'
6647:    +'anguage, hntContentEncoding, hntTransferEncoding, hntDate, hntServer, hntU'
6648:    +'serAgent, hntLocation, hntConnection, hntExpires, hntCacheControl, hntSetC'
6649:    +'ookie, hntCookie, hntAuthorization, hntVia, hntWarning, hntContentRange, h'
6650:    +'ntXForwardedFor, hntPragma, hntXPoweredBy, hntWWWAuthenticate, hntLastModi'
6651:    +'fied, hntETag, hntProxyAuthorization, hntReferer, hntAge, hntAcceptRanges,'
6652:    +' hntAcceptEncoding, hntAcceptLanguage, hntAcceptCharset, hntIfModifiedSinc'
6653:    +'e, hntIfUnmodifiedSince, hntRetryAfter, hntUpgrade, hntStatus, hntProxyCon'
6654:    +'nection, hntOrigin, hntKeepAlive )
6655:   THTTPHeaderName', 'record Value : THTTPHeaderNameEnum; Custom: AnsiString; end
6656:   THTTPCustomHeader', 'record FieldName : AnsiString; FieldValue :'
6657:    +' AnsiString; end
6658:   //PHTTPCustomHeader', '^THTTPCustomHeader // will not work
6659:   THTTPContentLengthEnum', '( hcltNone, hcltByteCount )
6660:   THTTPContentLength', 'record Value : THTTPContentLengthEnum; ByteCount:Int64; end
```

```
6661:   //PHTTPContentLength', '^THTTPContentLength // will not work
6662:   THTTPContentTypeMajor', '( hctmCustom, hctmText, hctmImage )
6663:   THTTPContentTypeEnum', '( hctNone, hctCustomParts, hctCustomStri'
6664:    +'ng, hctTextHtml, hctTextAscii, hctTextCss, hctTextPlain, hctTextXml, hctTe'
6665:    +'xtCustom, hctImageJpeg, hctImagePng, hctImageGif, hctImageCustom, hctAppli'
6666:    +'cationJSON, hctApplicationOctetStream, hctApplicationJavaScript, hctApplic'
6667:    +'ationCustom, hctAudioCustom, hctVideoCustom )
6668:   THTTPContentType', 'record Value : THTTPContentTypeEnum; CustomM'
6669:    +'ajor : AnsiString; CustomMinor : AnsiString; Parameters : AnsiStringArray;'
6670:    +' CustomStr : AnsiString; end
6671:   THTTPDateFieldEnum', '( hdNone, hdCustom, hdParts, hdDateTime )
6672:   THTTPDateField', 'record Value : THTTPDateFieldEnum; DayOfWeek :'
6673:    +' Integer; Day : integer; Month : integer; Year : Integer; Hour : integer; '
6674:    +'Min : integer; Sec : Integer; TimeZoneGMT : Boolean; CustomTimeZone : Ansi'
6675:    +'String; DateTime : TDateTime; Custom : AnsiString; end
6676:   THTTPTransferEncodingEnum', '( hteNone, hteCustom, hteChunked )
6677:   THTTPTransferEncoding', 'record Value : THTTPTransferEncodingEnu'
6678:    +'m; Custom : AnsiString; end
6679:   THTTPConnectionFieldEnum', '( hcfNone, hcfCustom, hcfClose, hcfKeepAlive )
6680:   THTTPConnectionField', 'record Value : THTTPConnectionFieldEnum;'
6681:    +' Custom : AnsiString; end
6682:   THTTPAgeFieldEnum', '( hafNone, hafCustom, hafAge )
6683:   THTTPAgeField', 'record Value: THTTPAgeFieldEnum; Age : Int64;Custom:AnsiString; end
6684:   THTTPCacheControlFieldEnum', '( hccfNone, hccfDecoded, hccfCustom )
6685:   THTTPCacheControlRequestSubField', '( hccsfNoCache, hccsfNoStore'
6686:    +', hccsfMaxAge, hccsfMaxStale, hccsfMinFresh, hccsfNoTransform, hccsfOnlyIfCached )
6687:   THTTPCacheControlResponseSubField', '( hccrfPublic, hccrfPrivate'
6688:    +', hccrfNoCache, hccrfNoStore, hccrfNoTransform, hccrfMustRevalidate, hccrf'
6689:    +'ProxyRevalidate, hccrfMaxAge, hccrfSMaxAge )
6690:   THTTPCacheControlField', 'record Value : THTTPCacheControlFieldEnum; end
6691:   THTTPContentEncodingEnum', '( hceNone, hceCustom, hceIdentity, h'
6692:    +'ceCompress, hceDeflate, hceExi, hceGzip, hcePack200Gzip )
6693:   THTTPContentEncoding', 'record Value:THTTPContentEncodingEnum;Custom:AnsiString; end;
6694:   THTTPContentEncodingFieldEnum', '( hcefNone, hcefList )
6695:   THTTPContentEncodingField', 'record Value : THTTPContentEncoding'
6696:    +'FieldEnum; List : array of THTTPContentEncoding; end
6697:   THTTPRetryAfterFieldEnum', '( hrafNone, hrafCustom, harfDate, harfSeconds )
6698:   THTTPRetryAfterField', 'record Value : THTTPRetryAfterFieldEnum;'
6699:    +' Custom : AnsiString; Date : TDateTime; Seconds : Int64; end
6700:   THTTPContentRangeFieldEnum', '( hcrfNone, hcrfCustom, hcrfByteRange )
6701:   THTTPContentRangeField', 'record Value : THTTPContentRangeFieldE'
6702:    +'num; ByteFirst : Int64; ByteLast : Int64; ByteSize : Int64; Custom : AnsiString; end
6703:   THTTPSetCookieFieldEnum', '( hscoNone, hscoDecoded, hscoCustom )
6704:   THTTPSetCookieCustomField', 'record Name : AnsiString; Value : AnsiString; end
6705:   THTTPSetCookieCustomFieldArray', 'array of THTTPSetCookieCustomField
6706:   THTTPSetCookieField', 'record Value : THTTPSetCookieFieldEnum; D'
6707:    +'omain : AnsiString; Path : AnsiString; Expires : THTTPDateField; MaxAge : '
6708:    +'Int64; HttpOnly : Boolean; Secure : Boolean; CustomFields : THTTPSetCookie'
6709:    +'CustomFieldArray; Custom : AnsiString; end
6710:   //PHTTPSetCookieField', '^THTTPSetCookieField // will not work
6711:   THTTPSetCookieFieldArray', 'array of THTTPSetCookieField
6712:   THTTPCookieFieldEnum', '( hcoNone, hcoDecoded, hcoCustom )
6713:   THTTPCookieFieldEntry', 'record Name : AnsiString; Value : AnsiString; end
6714:   //PHTTPCookieFieldEntry', '^THTTPCookieFieldEntry // will not work
6715:   THTTPCookieFieldEntryArray', 'array of THTTPCookieFieldEntry
6716:   THTTPCookieField', 'record Value : THTTPCookieFieldEnum; Entries'
6717:    +' : THTTPCookieFieldEntryArray; Custom : AnsiString; end
6718:   THTTPCommonHeaders', 'record TransferEncoding : THTTPTransferEnc'
6719:    +'oding; ContentType : THTTPContentType; ContentLength : THTTPContentLength;'
6720:    +' Connection : THTTPConnectionField; ProxyConnection : THTTPConnectionField'
6721:    +'; Date : THTTPDateField; ContentEncoding : THTTPContentEncodingField; end
6722:   THTTPCustomHeaders', 'array of THTTPCustomHeader
6723:    //THTTPFixedHeaders','array[THTTPHeaderNameEnum] of AnsiString
6724:   THTTPFixedHeaders','array[0..42] of AnsiString
6725:    THTTPMethodEnum', '( hmNone, hmCustom, hmGET, hmPUT, hmPOST, hmC'
6726:    +'ONNECT, hmHEAD, hmDELETE, hmOPTIONS, hmTRACE )
6727:   THTTPMethod', 'record Value : THTTPMethodEnum; Custom : AnsiString; end
6728:   THTTPRequestStartLine', 'record Method : THTTPMethod; URI : Ansi'
6729:    +'String; Version : THTTPVersion; end
6730:   THTTPRequestHeader', 'record CommonHeaders : THTTPCommonHeaders;'
6731:    +' FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Coo'
6732:    +'kie : THTTPCookieField; IfModifiedSince : THTTPDateField; IfUnmodifiedSinc'
6733:    +'e : THTTPDateField; end
6734:   //PHTTPRequestHeader', '^THTTPRequestHeader // will not work
6735:   THTTPRequest', 'record StartLine : THTTPRequestStartLine; Header'
6736:    +' : THTTPRequestHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6737:   THTTPResponseStartLineMessage', '( hslmNone, hslmCustom, hslmOK)
6738:   THTTPResponseStartLine', 'record Version : THTTPVersion; Code : '
6739:    +'Integer; Msg : THTTPResponseStartLineMessage; CustomMsg : AnsiString; end
6740:   THTTPResponseHeader', 'record CommonHeaders : THTTPCommonHeaders'
6741:    +'; FixedHeaders : THTTPFixedHeaders; CustomHeaders : THTTPCustomHeaders; Co'
6742:    +'okies : THTTPSetCookieFieldArray; Expires : THTTPDateField; LastModified :'
6743:    +' THTTPDateField; Age : THTTPAgeField; end
6744:   //PHTTPResponseHeader', '^THTTPResponseHeader // will not work
6745:   THTTPResponse', 'record StartLine : THTTPResponseStartLine; Head'
6746:    +'er : THTTPResponseHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6747:  Function HTTPMessageHasContent( const H : THTTPCommonHeaders) : Boolean
6748:  Procedure InitHTTPRequest( var A : THTTPRequest)
6749:  Procedure InitHTTPResponse( var A : THTTPResponse)
```

```
6750:  Procedure ClearHTTPVersion( var A : THTTPVersion)
6751:  Procedure ClearHTTPContentLength( var A : THTTPContentLength)
6752:  Procedure ClearHTTPContentType( var A : THTTPContentType)
6753:  Procedure ClearHTTPDateField( var A : THTTPDateField)
6754:  Procedure ClearHTTPTransferEncoding( var A : THTTPTransferEncoding)
6755:  Procedure ClearHTTPConnectionField( var A : THTTPConnectionField)
6756:  Procedure ClearHTTPAgeField( var A : THTTPAgeField)
6757:  Procedure ClearHTTPContentEncoding( var A : THTTPContentEncoding)
6758:  Procedure ClearHTTPContentEncodingField( var A : THTTPContentEncodingField)
6759:  Procedure ClearHTTPContentRangeField( var A : THTTPContentRangeField)
6760:  Procedure ClearHTTPSetCookieField( var A : THTTPSetCookieField)
6761:  Procedure ClearHTTPCommonHeaders( var A : THTTPCommonHeaders)
6762:  //Procedure ClearHTTPFixedHeaders( var A : THTTPFixedHeaders)
6763:  Procedure ClearHTTPCustomHeaders( var A : THTTPCustomHeaders)
6764:  Procedure ClearHTTPCookieField( var A : THTTPCookieField)
6765:  Procedure ClearHTTPMethod( var A : THTTPMethod)
6766:  Procedure ClearHTTPRequestStartLine( var A : THTTPRequestStartLine)
6767:  Procedure ClearHTTPRequestHeader( var A : THTTPRequestHeader)
6768:  Procedure ClearHTTPRequest( var A : THTTPRequest)
6769:  Procedure ClearHTTPResponseStartLine( var A : THTTPResponseStartLine)
6770:  Procedure ClearHTTPResponseHeader( var A : THTTPResponseHeader)
6771:  Procedure ClearHTTPResponse( var A : THTTPResponse)
6772:   THTTPStringOption', '( hsoNone )
6773:   THTTPStringOptions', 'set of THTTPStringOption
6774:   FindClass('TOBJECT'),'TAnsiStringBuilder
6775:
6776:  Procedure BuildStrHTTPVersion(const A:THTTPVersion; const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6777:  Procedure BuildStrHTTPContentLengthValue( const A : THTTPContentLength; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6778:  Procedure BuildStrHTTPContentLength( const A : THTTPContentLength; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6779:  Procedure BuildStrHTTPContentTypeValue( const A : THTTPContentType; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6780:  Procedure BuildStrHTTPContentType(const A:THTTPContentType;const B:TAnsiStringBuilder; const
      P:THTTPStringOptions)
6781:  Procedure BuildStrRFCDateTime( const DOW, Da, Mo, Ye, Ho, Mi, Se : Integer; const TZ : AnsiString; const
      B : TAnsiStringBuilder; const P : THTTPStringOptions)
6782:  Procedure BuildStrHTTPDateFieldValue( const A : THTTPDateField; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
6783:  Procedure BuildStrHTTPDateField(const A:THTTPDateField;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6784:  Procedure BuildStrHTTPTransferEncodingValue( const A : THTTPTransferEncoding; const B :
      TAnsiStringBuilder; const P : THTTPStringOptions)
6785:  Procedure BuildStrHTTPTransferEncoding( const A : THTTPTransferEncoding; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6786:  Procedure BuildStrHTTPContentRangeField( const A : THTTPContentRangeField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6787:  Procedure BuildStrHTTPConnectionFieldValue( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6788:  Procedure BuildStrHTTPConnectionField( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6789:  Procedure BuildStrHTTPAgeField(const A:THTTPAgeField;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6790:  Procedure BuildStrHTTPContentEncoding( const A : THTTPContentEncoding; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6791:  Procedure BuildStrHTTPContentEncodingField(const A:THTTPContentEncodingField;const
      B:TAnsiStringBuilder;const P:THTTPStringOptions)
6792:  Procedure BuildStrHTTPProxyConnectionField(const A : THTTPConnectionField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6793:  Procedure BuildStrHTTPCommonHeaders( const A : THTTPCommonHeaders; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6794:  Procedure BuildStrHTTPFixedHeaders(const A:THTTPFixedHeaders;const B:TAnsiStrBuilder;const
      P:THTTPStringOptions)
6795:  Procedure BuildStrHTTPCustomHeaders( const A : THTTPCustomHeaders; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6796:  Procedure BuildStrHTTPSetCookieFieldValue( const A : THTTPSetCookieField; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6797:  Procedure BuildStrHTTPCookieFieldValue( const A : THTTPCookieField; const B : TAnsiStringBuilder; const P
      : THTTPStringOptions)
6798:  Procedure BuildStrHTTPCookieField(const A:THTTPCookieField;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6799:  Procedure BuildStrHTTPMethod( const A : THTTPMethod; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
6800:  Procedure BuildStrHTTPRequestStartLine( const A : THTTPRequestStartLine; const B : TAnsiStringBuilder;
      const P : THTTPStringOptions)
6801:  Procedure BuildStrHTTPRequestHeader(const A:THTTPRequestHeader;const B:TAnsiStringBuilder;const
      P:THTTPStringOptions);
6802:  Procedure BuildStrHTTPRequest( const A : THTTPRequest; const B : TAnsiStringBuilder; const P :
      THTTPStringOptions)
6803:  Procedure BuildStrHTTPResponseCookieFieldArray( const A : THTTPSetCookieFieldArray; const B :
      TAnsiStringBuilder; const P : THTTPStringOptions)
6804:  Procedure BuildStrHTTPResponseStartLine(const A:THTTPResponseStartLine;const B:TAnsiStrBldr;const P
      THTTPStrOptions);
6805:  Procedure BuildStrHTTPResponseHeader(const A:THTTPRespHeader;const B:TAnsiStrBuilder;const
      P:THTTPStringOptions);
6806:  Procedure BuildStrHTTPResponse(const A:THTTPResponse; const B:TAnsiStringBuilder; const
      P:THTTPStringOptions);
6807:  Function HTTPContentTypeValueToStr( const A : THTTPContentType) : AnsiString
```

```
6808:  Function HTTPSetCookieFieldValueToStr( const A : THTTPSetCookieField) : AnsiString
6809:  Function HTTPCookieFieldValueToStr( const A : THTTPCookieField) : AnsiString
6810:  Function HTTPMethodToStr( const A : THTTPMethod) : AnsiString
6811:  Function HTTPRequestToStr( const A : THTTPRequest) : AnsiString
6812:  Function HTTPResponseToStr( const A : THTTPResponse) : AnsiString
6813:  Procedure PrepareCookie( var A:THTTPCookieField;const B:THTTPSetCookieFieldArray; const
      Domain:AnsiString; const Secure : Boolean);
6814:   THTTPParserHeaderParseFunc', 'Function ( const HeaderName : THTT'
6815:    +'PHeaderNameEnum; const HeaderPtr : ___Pointer) : Boolean
6816:   SIRegister_THTTPParser(CL);
6817:   FindClass('TOBJECT'),'THTTPContentDecoder
6818:   THTTPContentDecoderProc', 'Procedure ( const Sender : THTTPContentDecoder)
6819:   THTTPContentDecoderContentType', '( crctFixedSize, crctChunked, crctUnsized )
6820:   THTTPContentDecoderChunkState', '( crcsChunkHeader, crcsContent,'
6821:    +' crcsContentCRLF, crcsTrailer, crcsFinished )
6822:   THTTPContentDecoderLogEvent', 'Procedure ( const Sender : THTTPContentDecoder; const LogMsg : String)
6823:   SIRegister_THTTPContentDecoder(CL);
6824:   THTTPContentReaderMechanism', '( hcrmEvent, hcrmString, hcrmStream, hcrmFile )
6825:   FindClass('TOBJECT'),'THTTPContentReader
6826:   THTTPContentReaderProc', 'Procedure ( const Sender : THTTPContentReader)
6827:   THTTPContentReaderLogEvent','Procedure(const Sender:THTTPContentReader;const LogMsg:String;const
      LogLevel:Int;
6828:   SIRegister_THTTPContentReader(CL);
6829:   THTTPContentWriterMechanism', '( hctmEvent, hctmString, hctmStream, hctmFile )
6830:   FindClass('TOBJECT'),'THTTPContentWriter
6831:   THTTPContentWriterLogEvent', 'Procedure (const Sender : THTTPContentWriter;const LogMsg:AnsiString);
6832:   SIRegister_THTTPContentWriter(CL);
6833:  Procedure SelfTestcHTTPUtils
6834: end;
6835:
6836: (*----------------------------------------------------------------------------*)
6837: procedure SIRegister_cTLSUtils(CL: TPSPascalCompiler);
6838: begin
6839:  'TLSLibraryVersion','String').SetString( '1.00
6840:  'TLSError_None','LongInt').SetInt( 0);
6841:  'TLSError_InvalidBuffer','LongInt').SetInt( 1);
6842:  'TLSError_InvalidParameter','LongInt').SetInt( 2);
6843:  'TLSError_InvalidCertificate','LongInt').SetInt( 3);
6844:  'TLSError_InvalidState','LongInt').SetInt( 4);
6845:  'TLSError_DecodeError','LongInt').SetInt( 5);
6846:  'TLSError_BadProtocol','LongInt').SetInt( 6);
6847:  Function TLSErrorMessage( const TLSError : Integer) : String
6848:   SIRegister_ETLSError(CL);
6849:   TTLSProtocolVersion', 'record major : Byte; minor : Byte; end
6850:   PTLSProtocolVersion', '^TTLSProtocolVersion // will not work
6851:  Procedure InitSSLProtocolVersion30( var A : TTLSProtocolVersion)
6852:  Procedure InitTLSProtocolVersion10( var A : TTLSProtocolVersion)
6853:  Procedure InitTLSProtocolVersion11( var A : TTLSProtocolVersion)
6854:  Procedure InitTLSProtocolVersion12( var A : TTLSProtocolVersion)
6855:  Function IsTLSProtocolVersion( const A, B : TTLSProtocolVersion) : Boolean
6856:  Function IsSSL2( const A : TTLSProtocolVersion) : Boolean
6857:  Function IsSSL3( const A : TTLSProtocolVersion) : Boolean
6858:  Function IsTLS10( const A : TTLSProtocolVersion) : Boolean
6859:  Function IsTLS11( const A : TTLSProtocolVersion) : Boolean
6860:  Function IsTLS12( const A : TTLSProtocolVersion) : Boolean
6861:  Function IsTLS10OrLater( const A : TTLSProtocolVersion) : Boolean
6862:  Function IsTLS11OrLater( const A : TTLSProtocolVersion) : Boolean
6863:  Function IsTLS12OrLater( const A : TTLSProtocolVersion) : Boolean
6864:  Function IsFutureTLSVersion( const A : TTLSProtocolVersion) : Boolean
6865:  Function IsKnownTLSVersion( const A : TTLSProtocolVersion) : Boolean
6866:  Function TLSProtocolVersionToStr( const A : TTLSProtocolVersion) : String
6867:  Function TLSProtocolVersionName( const A : TTLSProtocolVersion) : String
6868:   PTLSRandom', '^TTLSRandom // will not work
6869:  Procedure InitTLSRandom( var Random : TTLSRandom)
6870:  Function TLSRandomToStr( const Random : TTLSRandom) : AnsiString
6871:  'TLSSessionIDMaxLen','LongInt').SetInt( 32);
6872:  Procedure InitTLSSessionID( var SessionID : TTLSSessionID; const A : AnsiString)
6873:  Function EncodeTLSSessionID(var Buffer:string;const Size:Int;const SessionID:TTLSSessionID):Int;
6874:  Function DecodeTLSSessionID(const Buffer:string;const Size:Int;var SessionID:TTLSSessionID):Int;
6875:   TTLSSignatureAndHashAlgorithm', 'record Hash : TTLSHashAlgorithm'
6876:    +'; Signature : TTLSSignatureAlgorithm; end
6877:  // PTLSSignatureAndHashAlgorithm', '^TTLSSignatureAndHashAlgorithm +'// will not work
6878:   TTLSSignatureAndHashAlgorithmArray', 'array of TTLSSignatureAndHashAlgorithm
6879:   TTLSKeyExchangeAlgorithm', '( tlskeaNone, tlskeaNULL, tlskeaDHE_'
6880:    +'DSS, tlskeaDHE_RSA, tlskeaDH_Anon, tlskeaRSA, tlskeaDH_DSS, tlskeaDH_RSA )
6881:   TTLSMACAlgorithm', '( tlsmaNone, tlsmaNULL, tlsmaHMAC_MD5, tlsma'
6882:    +'HMAC_SHA1, tlsmaHMAC_SHA256, tlsmaHMAC_SHA384, tlsmaHMAC_SHA512 )
6883:   TTLSMacAlgorithmInfo', 'record Name : AnsiString; DigestSize : I'
6884:    +'nteger; Supported : Boolean; end
6885:   PTLSMacAlgorithmInfo', '^TTLSMacAlgorithmInfo // will not work
6886:  'TLS_MAC_MAXDIGESTSIZE','LongInt').SetInt( 64);
6887:   TTLSPRFAlgorithm', '( tlspaSHA256 )
6888:  Function tlsP_MD5( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6889:  Function tlsP_SHA1( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6890:  Function tlsP_SHA256( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6891:  Function tlsP_SHA512( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6892:  Function tls10PRF( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6893:  Function tls12PRF_SHA256( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6894:  Function tls12PRF_SHA512( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
```

```
6895:  Function TLSPRF(const ProtoVersion:TTLSProtocolVersion;const Secret,ALabel,Seed:AString;const
       Size:Int):AString;
6896:  Function tls10KeyBlock(const MasterSecret, ServerRandom,ClientRandom:AnsiString; const
       Size:Integer):AnsiString
6897:  Function tls12SHA256KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
       Size:Int):AnsiString;
6898:  Function tls12SHA512KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
       Size:Int):AnsiString;
6899:  Function TLSKeyBlock( const ProtocolVersion : TTLSProtocolVersion; const MasterSecret, ServerRandom,
       ClientRandom : AnsiString; const Size : Integer) : AnsiString
6900:  Function tls10MasterSecret(const PreMasterSecret,ClientRandom, ServerRandom:AnsiString) :AnsiString;
6901:  Function tls12SHA256MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString):AnsiString;
6902:  Function tls12SHA512MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString): AnsiString;
6903:  Function TLSMasterSecret( const ProtocolVersion: TTLSProtocolVersion;const PreMasterSecret,ClientRandom,
       ServerRandom:AnsiString) : AnsiString
6904:    TTLSKeys', 'record KeyBlock : AnsiString; ClientMACKey : AnsiStr'
6905:     +'ing; ServerMACKey : AnsiString; ClientEncKey : AnsiString; ServerEncKey : '
6906:     +'AnsiString; ClientIV : AnsiString; ServerIV : AnsiString; end
6907:  Procedure GenerateTLSKeys( const ProtocolVersion : TTLSProtocolVersion; const MACKeyBits, CipherKeyBits,
       IVBits : Integer; const MasterSecret, ServerRandom, ClientRandom : AnsiString; var TLSKeys : TTLSKeys)
6908:  Procedure GenerateFinalTLSKeys( const ProtocolVersion : TTLSProtocolVersion; const IsExportable :
       Boolean; const ExpandedKeyBits : Integer; const ServerRandom, ClientRandom : AnsiString; var TLSKeys :
       TTLSKeys)
6909:  'TLS_PLAINTEXT_FRAGMENT_MAXSIZE','LongInt').SetInt( 16384 - 1);
6910:  'TLS_COMPRESSED_FRAGMENT_MAXSIZE','LongInt').SetInt( 16384 + 1024);
6911:  Procedure SelfTestcTLSUtils
6912: end;
6913:
6914: (*----------------------------------------------------------------------------*)
6915: procedure SIRegister_Reversi(CL: TPSPascalCompiler);
6916: begin
6917:   sPosData', 'record corner : boolean; square2x2 : boolean; edge :'
6918:    +' boolean; stable : integer; internal : integer; disks : integer; mx : integer; my : integer; end
6919:  // pBoard', '^tBoard // will not work
6920:  Function rCalculateData( cc : byte; cx, cy : integer) : sPosData
6921:  Function rCheckMove( color : byte; cx, cy : integer) : integer
6922:  //Function rDoStep( data : pBoard) : word
6923:  Function winExecAndWait( const sAppPath : string; wVisible : word) : boolean
6924: end;
6925:
6926: procedure SIRegister_IWDBCommon(CL: TPSPascalCompiler);
6927: begin
6928: Function InEditMode( ADataset : TDataset) : Boolean
6929: Function CheckDataSource( ADataSource : TDataSource) : Boolean;
6930: Function CheckDataSource1(ADataSource:TDataSource;const AFieldName:string;var VField:TField):boolean;
6931: Function GetFieldText( AField : TField) : String
6932: end;
6933:
6934: procedure SIRegister_SortGrid(CL: TPSPascalCompiler);
6935: begin
6936:   TPrintMode', '( pmPrint, pmPreview, pmPageCount )
6937:   TMyPrintRange', '( prAll, prSelected )
6938:   TSortStyle', '( ssAutomatic, ssNormal, ssNumeric, ssNumericExten'
6939:    +'ded, ssDateTime, ssTime, ssCustom )
6940:   TSortDirection', '( sdAscending, sdDescending )
6941:   TSetChecked', 'Procedure ( Sender : TObject; ACol, ARow : integer; State : Boolean)
6942:   TGetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integ'
6943:    +'er; var Strs : TStringList; var Width, Height : integer; var Sorted : Boolean)
6944:   TSetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integer; Str : String)
6945:   TSetEllipsis', 'Procedure ( Sender : TObject; ACol, ARow : integer)
6946:   SIRegister_TSortOptions(CL);
6947:   SIRegister_TPrintOptions(CL);
6948:   TSortedListEntry', 'record Str : String; RowNum : integer; SortOption : TSortOptions; end
6949:   SIRegister_TSortedList(CL);
6950:   TCellBevelStyle', '( cbNone, cbRaised, cbLowered )
6951:   TCellBevel', 'record Style: TCellBevelStyle; UpperLeftColor: TColor; LowerRightColor : TColor; end
6952:   TVertAlignment', '( taTopJustify, taBottomJustify, taMiddle )
6953:   TFormatOptions', 'record Brush : TBrush; Font : TFont; Alignment'
6954:    +'Horz : TAlignment; AlignmentVert : TVertAlignment; Bevel : TCellBevel; HideText : Boolean; end
6955:   SIRegister_TFontSetting(CL);
6956:   SIRegister_TFontList(CL);
6957:   CL.AddTypeS(TFormatDrawCellEvent', 'Procedure ( Sender : TObject; Col, Row :'
6958:    + integer;State:TGridDrawState;var FormatOptions:TFormatOptions;var CheckBox,Combobox,Ellipsis:Bool);
6959:   TSetFilterEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
6960:   TSearchEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
6961:   TUpdateGridEvent', 'Procedure ( Sender : TObject; ARow : integer)
6962:   TSizeChangedEvent', 'Procedure ( Sender : TObject; OldColCount, OldRowCount : integer)
6963:   TBeginSortEvent', 'Procedure ( Sender : TObject; var Col : integer)
6964:   TEndSortEvent', 'Procedure ( Sender : TObject; Col : integer)
6965:   TGetSortStyleEvent', 'Procedure ( Sender : TObject; Col : intege'
6966:    +'r; var SortStyle : TSortStyle)
6967:   TCellValidateEvent', 'Procedure ( Sender : TObject; ACol, ARow :'
6968:    +' integer; const OldValue : string; var NewValue : String; var Valid : Boolean)
6969:   SIRegister_TSortGrid(CL);
6970:  Function ExtendedCompare( const Str1, Str2 : String) : Integer
6971:  Function NormalCompare( const Str1, Str2 : String) : Integer
6972:  Function DateTimeCompare( const Str1, Str2 : String) : Integer
6973:  Function NumericCompare( const Str1, Str2 : String) : Integer
6974:  Function TimeCompare( const Str1, Str2 : String) : Integer
```

```
6975:  //Function Compare( Item1, Item2 : Pointer) : Integer
6976: end;
6977:
6978: ********************************** procedure Register_IB(CL: TPSPascalCompiler);
6979:  Procedure IBAlloc( var P, OldSize, NewSize : Integer)
6980:  Procedure IBError( ErrMess : TIBClientError; const Args : array of const)
6981:  Procedure IBDataBaseError
6982:  Function StatusVector : PISC_STATUS
6983:  Function StatusVectorArray : PStatusVector
6984:  Function CheckStatusVector( ErrorCodes : array of ISC_STATUS) : Boolean
6985:  Function StatusVectorAsText : string
6986:  Procedure SetIBDataBaseErrorMessages( Value : TIBDataBaseErrorMessages)
6987:  Function GetIBDataBaseErrorMessages : TIBDataBaseErrorMessages
6988:
6989:
6990: //****************************unit uPSI_BoldUtils;************************************
6991:  Function CharCount( c : char; const s : string) : integer
6992:  Function BoldNamesEqual( const name1, name2 : string) : Boolean
6993:  Procedure BoldAppendToStrings(strings: TStrings; const aString: string; const ForceNewLine:Boolean)
6994:  Function BoldSeparateStringList(strings:TStringList;const Separator,PreString,PostString:String):String
6995:  Function BoldSeparatedAppend( const S1, S2 : string; const Separator : string) : string
6996:  Function BoldTrim( const S : string) : string
6997:  Function BoldIsPrefix( const S, Prefix : string) : Boolean
6998:  Function BoldStrEqual( P1, P2 : PChar; Len : integer) : Boolean
6999:  Function BoldStrAnsiEqual( P1, P2 : PChar; Len : integer) : Boolean
7000:  Function BoldAnsiEqual( const S1, S2 : string) : Boolean
7001:  Function BoldStrStringEqual( const S1 : string; P2 : PChar; Len : integer) : Boolean
7002:  Function BoldCaseIndependentPos( const Substr, S : string) : Integer
7003:  //Procedure EnumToStrings( aTypeInfo : pTypeInfo; Strings : TStrings)
7004:  Function CapitalisedToSpaced( Capitalised : String) : String
7005:  Function SpacedToCapitalised( Spaced : String) : String
7006:  Function BooleanToString( BoolValue : Boolean) : String
7007:  Function StringToBoolean( StrValue : String) : Boolean
7008:  Function GetUpperLimitForMultiplicity( const Multiplicity : String) : Integer
7009:  Function GetLowerLimitForMultiplicity( const Multiplicity : String) : Integer
7010:  Function StringListToVarArray( List : TStringList) : variant
7011:  Function IsLocalMachine( const Machinename : WideString) : Boolean
7012:  Function GetComputerNameStr : string
7013:  Function TimeStampComp( const Time1, Time2 : TTimeStamp) : Integer
7014:  Function BoldStrToDateFmt(const S:string;const DateFormat:string;const DateSeparatorChar:char):TDateTime
7015:  Function BoldDateToStrFmt(const aDate:TDateTime;DateFormat:string;const DateSeparatorChar:char):String;
7016:  Function BoldParseFormattedDateList(const value:String;const formats:TStrings;var Date:TDateTime):Boolean;
7017:  Function BoldParseFormattedDate(const value:String;const formats:array of string; var
      Date:TDateTime):Boolean;
7018:  Procedure EnsureTrailing( var Str : String; ch : char)
7019:  Function BoldDirectoryExists( const Name : string) : Boolean
7020:  Function BoldForceDirectories( Dir : string) : Boolean
7021:  Function BoldRootRegistryKey : string
7022:  Function GetModuleFileNameAsString( IncludePath : Boolean) : string
7023:  Function BoldVariantToStrings( V : OleVariant; Strings : TStrings) : Integer
7024:  Function LogicalAnd( A, B : Integer) : Boolean
7025:  record TByHandleFileInformation dwFileAttributes : DWORD; '
7026:    +'ftCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime '
7027:    +': TFileTime; dwVolumeSerialNumber : DWORD; nFileSizeHigh : DWORD; nFileSiz'
7028:    +'eLow : DWORD; nNumberOfLinks : DWORD; nFileIndexHigh : DWORD; nFileIndexLow : DWORD; end
7029:  Function GetFileInformationByHandle(hFile:THandle;var lpFileInformation:TByHandleFileInformation):BOOL;
7030:  Function IsFirstInstance : Boolean
7031:  Procedure ActivateFirst( AString : PChar)
7032:  Procedure ActivateFirstCommandLine
7033:  function MakeAckPkt(const BlockNumber: Word): string;
7034:  procedure SendError(UDPBase:TIdUDPBase;APeerIP:string;const APort:Int;const ErrNumber:Word;ErrorString:
      string);
7035:  procedure SendError(UDPClient: TIdUDPClient; const ErrNumber: Word; ErrorString: string); overload;
7036:  procedure SendError(UDPBase: TIdUDPBase; APeerIP: string; const APort: Integer;  E: Exception); overload;
7037:  procedure SendError(UDPClient: TIdUDPClient; E: Exception); overload;
7038:  function IdStrToWord(const Value: String): Word;
7039:  function IdWordToStr(const Value: Word): WordStr;
7040:  Function HasInstructionSet( const InstructionSet : TCPUInstructionSet) : Boolean
7041:  Function CPUFeatures : TCPUFeatures
7042:
7043:
7044: procedure SIRegister_xrtl_util_CPUUtils(CL: TPSPascalCompiler);
7045: begin
7046:   AddTypeS('TXRTLBitIndex', 'Integer
7047:  Function XRTLSwapBits( Data : Cardinal; Bit1Index, Bit2Index : TXRTLBitIndex) : Cardinal
7048:  Function XRTLBitTest( Data : Cardinal; BitIndex : TXRTLBitIndex) : Boolean
7049:  Function XRTLBitSet( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7050:  Function XRTLBitReset( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7051:  Function XRTLBitComplement( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7052:  Function XRTLSwapHiLo16( X : Word) : Word
7053:  Function XRTLSwapHiLo32( X : Cardinal) : Cardinal
7054:  Function XRTLSwapHiLo64( X : Int64) : Int64
7055:  Function XRTLROL32( A, S : Cardinal) : Cardinal
7056:  Function XRTLROR32( A, S : Cardinal) : Cardinal
7057:  Function XRTLROL16( A : Word; S : Cardinal) : Word
7058:  Function XRTLROR16( A : Word; S : Cardinal) : Word
7059:  Function XRTLROL8( A : Byte; S : Cardinal) : Byte
7060:  Function XRTLROR8( A : Byte; S : Cardinal) : Byte
7061:  //Procedure XRTLXorBlock( I1, I2, O1 : PByteArray; Len : integer)
```

```
7062:  //Procedure XRTLIncBlock( P : PByteArray; Len : integer)
7063:  Procedure XRTLUMul64( const A, B : Integer; var MulL, MulH : Integer)
7064:  Function XRTLPopulation( A : Cardinal) : Cardinal
7065: end;
7066:
7067: Function XRTLURLDecode( const ASrc : WideString) : WideString
7068: Function XRTLURLEncode( const ASrc : WideString) : WideString
7069: Function XRTLURINormalize( const AURI : WideString) : WideString
7070: Procedure XRTLURIParse(const AURI:WideString;var VProtocol,VHost,VPath,VDocument,VPort,VBookmark,VUserName,
      VPassword : WideString)
7071: Function XRTLExtractLongPathName(APath: string): string;
7072:
7073: procedure SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
7074: begin
7075:   AddTypeS('Int8', 'ShortInt
7076:   AddTypeS('Int16', 'SmallInt
7077:   AddTypeS('Int32', 'LongInt
7078:   AddTypeS('UInt8', 'Byte
7079:   AddTypeS('UInt16', 'Word
7080:   AddTypeS('UInt32', 'LongWord
7081:   AddTypeS('UInt64', 'Int64
7082:   AddTypeS('Word8', 'UInt8
7083:   AddTypeS('Word16', 'UInt16
7084:   AddTypeS('Word32', 'UInt32
7085:   AddTypeS('Word64', 'UInt64
7086:   AddTypeS('LargeInt', 'Int64
7087:   AddTypeS('NativeInt', 'Integer
7088:   AddTypeS('NativeUInt', 'Cardinal
7089:   Const('BitsPerByte','LongInt').SetInt( 8);
7090:   Const('BitsPerWord','LongInt').SetInt( 16);
7091:   Const('BitsPerLongWord','LongInt').SetInt( 32);
7092:  //Const('BitsPerCardinal','LongInt').SetInt( BytesPerCardinal * 8);
7093:  //Const('BitsPerNativeWord','LongInt').SetInt( BytesPerNativeWord * 8);
7094:  Function MinI( const A, B : Integer) : Integer
7095:  Function MaxI( const A, B : Integer) : Integer
7096:  Function MinC( const A, B : Cardinal) : Cardinal
7097:  Function MaxC( const A, B : Cardinal) : Cardinal
7098:  Function SumClipI( const A, I : Integer) : Integer
7099:  Function SumClipC( const A : Cardinal; const I : Integer) : Cardinal
7100:  Function InByteRange( const A : Int64) : Boolean
7101:  Function InWordRange( const A : Int64) : Boolean
7102:  Function InLongWordRange( const A : Int64) : Boolean
7103:  Function InShortIntRange( const A : Int64) : Boolean
7104:  Function InSmallIntRange( const A : Int64) : Boolean
7105:  Function InLongIntRange( const A : Int64) : Boolean
7106:   AddTypeS('Bool8', 'ByteBool
7107:   AddTypeS('Bool16', 'WordBool
7108:   AddTypeS('Bool32', 'LongBool
7109:   AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )
7110:   AddTypeS('TCompareResultSet', 'set of TCompareResult
7111:  Function ReverseCompareResult( const C : TCompareResult) : TCompareResult
7112:  Const('MinSingle','Single').setExtended( 1.5E-45);
7113:  Const('MaxSingle','Single').setExtended( 3.4E+38);
7114:  Const('MinDouble','Double').setExtended( 5.0E-324);
7115:  Const('MaxDouble','Double').setExtended( 1.7E+308);
7116:  Const('MinExtended','Extended').setExtended(3.4E-4932);
7117:  Const('MaxExtended','Extended').setExtended(1.1E+4932);
7118:  Const('MinCurrency','Currency').SetExtended( - 922337203685477.5807);
7119:  Const('MaxCurrency','Currency').SetExtended( 922337203685477.5807);
7120:  Function MinF( const A, B : Float) : Float
7121:  Function MaxF( const A, B : Float) : Float
7122:  Function ClipF( const Value : Float; const Low, High : Float) : Float
7123:  Function InSingleRange( const A : Float) : Boolean
7124:  Function InDoubleRange( const A : Float) : Boolean
7125:  Function InCurrencyRange( const A : Float) : Boolean;
7126:  Function InCurrencyRange1( const A : Int64) : Boolean;
7127:  Function FloatExponentBase2( const A : Extended; var Exponent : Integer) : Boolean
7128:  Function FloatExponentBase10( const A : Extended; var Exponent : Integer) : Boolean
7129:  Function FloatIsInfinity( const A : Extended) : Boolean
7130:  Function FloatIsNaN( const A : Extended) : Boolean
7131:  Const('SingleCompareDelta','Extended').setExtended( 1.0E-34);
7132:  Const('DoubleCompareDelta','Extended').setExtended( 1.0E-280);
7133:  Const('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
7134:  Const('DefaultCompareDelta','Extended').SetExtended( 1.0E-34);
7135:  Function FloatZero( const A : Float; const CompareDelta : Float) : Boolean
7136:  Function FloatOne( const A : Float; const CompareDelta : Float) : Boolean
7137:  Function FloatsEqual( const A, B : Float; const CompareDelta : Float) : Boolean
7138:  Function FloatsCompare( const A, B : Float; const CompareDelta : Float) : TCompareResult
7139:  Const('SingleCompareEpsilon','Extended').setExtended( 1.0E-5);
7140:  Const('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13);
7141:  Const('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17);
7142:  Const('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
7143:  Function ApproxEqual( const A, B : Extended; const CompareEpsilon : Double) : Boolean
7144:  Function ApproxCompare( const A, B : Extended; const CompareEpsilon : Double): TCompareResult
7145:  Function cClearBit( const Value, BitIndex : LongWord) : LongWord
7146:  Function cSetBit( const Value, BitIndex : LongWord) : LongWord
7147:  Function cIsBitSet( const Value, BitIndex : LongWord) : Boolean
7148:  Function cToggleBit( const Value, BitIndex : LongWord) : LongWord
7149:  Function cIsHighBitSet( const Value : LongWord) : Boolean
```

```
7150:  Function SetBitScanForward( const Value : LongWord) : Integer;
7151:  Function SetBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7152:  Function SetBitScanReverse( const Value : LongWord) : Integer;
7153:  Function SetBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7154:  Function ClearBitScanForward( const Value : LongWord) : Integer;
7155:  Function ClearBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7156:  Function ClearBitScanReverse( const Value : LongWord) : Integer;
7157:  Function ClearBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7158:  Function cReverseBits( const Value : LongWord) : LongWord;
7159:  Function cReverseBits1( const Value : LongWord; const BitCount : Integer) : LongWord;
7160:  Function cSwapEndian( const Value : LongWord) : LongWord
7161:  Function cTwosComplement( const Value : LongWord) : LongWord
7162:  Function RotateLeftBits16( const Value : Word; const Bits : Byte) : Word
7163:  Function RotateLeftBits32( const Value : LongWord; const Bits : Byte) : LongWord
7164:  Function RotateRightBits16( const Value : Word; const Bits : Byte) : Word
7165:  Function RotateRightBits32( const Value : LongWord; const Bits : Byte) : LongWord
7166:  Function cBitCount( const Value : LongWord) : LongWord
7167:  Function cIsPowerOfTwo( const Value : LongWord) : Boolean
7168:  Function LowBitMask( const HighBitIndex : LongWord) : LongWord
7169:  Function HighBitMask( const LowBitIndex : LongWord) : LongWord
7170:  Function RangeBitMask( const LowBitIndex, HighBitIndex : LongWord) : LongWord
7171:  Function SetBitRange( const Value: LongWord; const LowBitIndex, HighBitIndex: LongWord) : LongWord
7172:  Function ClearBitRange(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7173:  Function ToggleBitRange(const Value:LongWord; const LowBitIndex,HighBitIndex: LongWord) : LongWord
7174:  Function IsBitRangeSet(const Value: LongWord; const LowBitIndex,HighBitIndex : LongWord) : Boolean
7175:  Function IsBitRangeClear(const Value: LongWord; const LowBitIndex,HighBitIndex: LongWord): Boolean
7176: //  AddTypeS('CharSet', 'set of AnsiChar
7177:   AddTypeS('CharSet', 'set of Char       //!!!
7178:   AddTypeS('AnsiCharSet', 'TCharSet
7179:   AddTypeS('ByteSet', 'set of Byte
7180:   AddTypeS('AnsiChar', 'Char
7181:    // Function AsCharSet( const C : array of AnsiChar) : CharSet
7182:  Function AsByteSet( const C : array of Byte) : ByteSet
7183:  Procedure ComplementChar( var C : CharSet; const Ch : Char)
7184:  Procedure ClearCharSet( var C : CharSet)
7185:  Procedure FillCharSet( var C : CharSet)
7186:  Procedure ComplementCharSet( var C : CharSet)
7187:  Procedure AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7188:  Procedure Union( var DestSet : CharSet; const SourceSet : CharSet)
7189:  Procedure Difference( var DestSet : CharSet; const SourceSet : CharSet)
7190:  Procedure Intersection( var DestSet : CharSet; const SourceSet : CharSet)
7191:  Procedure XORCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7192:  Function IsSubSet( const A, B : CharSet) : Boolean
7193:  Function IsEqual( const A, B : CharSet) : Boolean
7194:  Function IsEmpty( const C : CharSet) : Boolean
7195:  Function IsComplete( const C : CharSet) : Boolean
7196:  Function cCharCount( const C : CharSet) : Integer
7197:  Procedure ConvertCaseInsensitive( var C : CharSet)
7198:  Function CaseInsensitiveCharSet( const C : CharSet) : CharSet
7199:  Function IntRangeLength( const Low, High : Integer) : Int64
7200:  Function IntRangeAdjacent( const Low1, High1, Low2, High2 : Integer) : Boolean
7201:  Function IntRangeOverlap( const Low1, High1, Low2, High2 : Integer) : Boolean
7202:  Function IntRangeHasElement( const Low, High, Element : Integer) : Boolean
7203:  Function IntRangeIncludeElement( var Low, High : Integer; const Element : Integer) : Boolean
7204:  Function IntRangeIncludeElementRange(var Low,High:Integer;const LowElement,HighElement:Integer):Boolean
7205:  Function CardRangeLength( const Low, High : Cardinal) : Int64
7206:  Function CardRangeAdjacent( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7207:  Function CardRangeOverlap( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7208:  Function CardRangeHasElement( const Low, High, Element : Cardinal) : Boolean
7209:  Function CardRangeIncludeElement( var Low, High : Cardinal; const Element : Cardinal) : Boolean
7210:  Function CardRangeIncludeElementRange(var Low,High:Card;const LowElement,HighElement:Card):Boolean;
7211:   AddTypeS('UnicodeChar', 'WideChar
7212:  Function Compare( const I1, I2 : Boolean) : TCompareResult;
7213:  Function Compare1( const I1, I2 : Integer) : TCompareResult;
7214:  Function Compare2( const I1, I2 : Int64) : TCompareResult;
7215:  Function Compare3( const I1, I2 : Extended) : TCompareResult;
7216:  Function CompareA( const I1, I2 : AnsiString) : TCompareResult
7217:  Function CompareW( const I1, I2 : WideString) : TCompareResult
7218:  Function cSgn( const A : LongInt) : Integer;
7219:  Function cSgn1( const A : Int64) : Integer;
7220:  Function cSgn2( const A : Extended) : Integer;
7221:  AddTypeS('TConvertResult', '( convertOK, convertFormatError, convertOverflow )
7222:  Function AnsiCharToInt( const A : AnsiChar) : Integer
7223:  Function WideCharToInt( const A : WideChar) : Integer
7224:  Function CharToInt( const A : Char) : Integer
7225:  Function IntToAnsiChar( const A : Integer) : AnsiChar
7226:  Function IntToWideChar( const A : Integer) : WideChar
7227:  Function IntToChar( const A : Integer) : Char
7228:  Function IsHexAnsiChar( const Ch : AnsiChar) : Boolean
7229:  Function IsHexWideChar( const Ch : WideChar) : Boolean
7230:  Function IsHexChar( const Ch : Char) : Boolean
7231:  Function HexAnsiCharToInt( const A : AnsiChar) : Integer
7232:  Function HexWideCharToInt( const A : WideChar) : Integer
7233:  Function HexCharToInt( const A : Char) : Integer
7234:  Function IntToUpperHexAnsiChar( const A : Integer) : AnsiChar
7235:  Function IntToUpperHexWideChar( const A : Integer) : WideChar
7236:  Function IntToUpperHexChar( const A : Integer) : Char
7237:  Function IntToLowerHexAnsiChar( const A : Integer) : AnsiChar
7238:  Function IntToLowerHexWideChar( const A : Integer) : WideChar
```

```
7239:  Function IntToLowerHexChar( const A : Integer) : Char
7240:  Function IntToStringA( const A : Int64) : AnsiString
7241:  Function IntToStringW( const A : Int64) : WideString
7242:  Function IntToString( const A : Int64) : String
7243:  Function UIntToStringA( const A : NativeUInt) : AnsiString
7244:  Function UIntToStringW( const A : NativeUInt) : WideString
7245:  Function UIntToString( const A : NativeUInt) : String
7246:  Function LongWordToStrA( const A : LongWord; const Digits : Integer) : AnsiString
7247:  Function LongWordToStrW( const A : LongWord; const Digits : Integer) : WideString
7248:  Function LongWordToStrU( const A : LongWord; const Digits : Integer) : UnicodeString
7249:  Function LongWordToStr( const A : LongWord; const Digits : Integer) : String
7250:  Function LongWordToHexA(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):AnsiString;
7251:  Function LongWordToHexW(const A:LongWord;const Digits:Integer;const UpperCase:Boolean):WideString;
7252:  Function LongWordToHex( const A : LongWord; const Digits : Integer;const UpperCase:Boolean):String
7253:  Function LongWordToOctA( const A : LongWord; const Digits : Integer) : AnsiString
7254:  Function LongWordToOctW( const A : LongWord; const Digits : Integer) : WideString
7255:  Function LongWordToOct( const A : LongWord; const Digits : Integer) : String
7256:  Function LongWordToBinA( const A : LongWord; const Digits : Integer) : AnsiString
7257:  Function LongWordToBinW( const A : LongWord; const Digits : Integer) : WideString
7258:  Function LongWordToBin( const A : LongWord; const Digits : Integer) : String
7259:  Function TryStringToInt64A( const S : AnsiString; out A : Int64) : Boolean
7260:  Function TryStringToInt64W( const S : WideString; out A : Int64) : Boolean
7261:  Function TryStringToInt64( const S : String; out A : Int64) : Boolean
7262:  Function StringToInt64DefA( const S : AnsiString; const Default : Int64) : Int64
7263:  Function StringToInt64DefW( const S : WideString; const Default : Int64) : Int64
7264:  Function StringToInt64Def( const S : String; const Default : Int64) : Int64
7265:  Function StringToInt64A( const S : AnsiString) : Int64
7266:  Function StringToInt64W( const S : WideString) : Int64
7267:  Function StringToInt64( const S : String) : Int64
7268:  Function TryStringToIntA( const S : AnsiString; out A : Integer) : Boolean
7269:  Function TryStringToIntW( const S : WideString; out A : Integer) : Boolean
7270:  Function TryStringToInt( const S : String; out A : Integer) : Boolean
7271:  Function StringToIntDefA( const S : AnsiString; const Default : Integer) : Integer
7272:  Function StringToIntDefW( const S : WideString; const Default : Integer) : Integer
7273:  Function StringToIntDef( const S : String; const Default : Integer) : Integer
7274:  Function StringToIntA( const S : AnsiString) : Integer
7275:  Function StringToIntW( const S : WideString) : Integer
7276:  Function StringToInt( const S : String) : Integer
7277:  Function TryStringToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7278:  Function TryStringToLongWordW( const S : WideString; out A : LongWord) : Boolean
7279:  Function TryStringToLongWord( const S : String; out A : LongWord) : Boolean
7280:  Function StringToLongWordA( const S : AnsiString) : LongWord
7281:  Function StringToLongWordW( const S : WideString) : LongWord
7282:  Function StringToLongWord( const S : String) : LongWord
7283:  Function HexToUIntA( const S : AnsiString) : NativeUInt
7284:  Function HexToUIntW( const S : WideString) : NativeUInt
7285:  Function HexToUInt( const S : String) : NativeUInt
7286:  Function TryHexToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7287:  Function TryHexToLongWordW( const S : WideString; out A : LongWord) : Boolean
7288:  Function TryHexToLongWord( const S : String; out A : LongWord) : Boolean
7289:  Function HexToLongWordA( const S : AnsiString) : LongWord
7290:  Function HexToLongWordW( const S : WideString) : LongWord
7291:  Function HexToLongWord( const S : String) : LongWord
7292:  Function TryOctToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7293:  Function TryOctToLongWordW( const S : WideString; out A : LongWord) : Boolean
7294:  Function TryOctToLongWord( const S : String; out A : LongWord) : Boolean
7295:  Function OctToLongWordA( const S : AnsiString) : LongWord
7296:  Function OctToLongWordW( const S : WideString) : LongWord
7297:  Function OctToLongWord( const S : String) : LongWord
7298:  Function TryBinToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7299:  Function TryBinToLongWordW( const S : WideString; out A : LongWord) : Boolean
7300:  Function TryBinToLongWord( const S : String; out A : LongWord) : Boolean
7301:  Function BinToLongWordA( const S : AnsiString) : LongWord
7302:  Function BinToLongWordW( const S : WideString) : LongWord
7303:  Function BinToLongWord( const S : String) : LongWord
7304:  Function FloatToStringA( const A : Extended) : AnsiString
7305:  Function FloatToStringW( const A : Extended) : WideString
7306:  Function FloatToString( const A : Extended) : String
7307:  Function TryStringToFloatA( const A : AnsiString; out B : Extended) : Boolean
7308:  Function TryStringToFloatW( const A : WideString; out B : Extended) : Boolean
7309:  Function TryStringToFloat( const A : String; out B : Extended) : Boolean
7310:  Function StringToFloatA( const A : AnsiString) : Extended
7311:  Function StringToFloatW( const A : WideString) : Extended
7312:  Function StringToFloat( const A : String) : Extended
7313:  Function StringToFloatDefA( const A : AnsiString; const Default : Extended) : Extended
7314:  Function StringToFloatDefW( const A : WideString; const Default : Extended) : Extended
7315:  Function StringToFloatDef( const A : String; const Default : Extended) : Extended
7316:  Function EncodeBase64(const S,Alphabet:AnsiString; const Pad:Boolean; const PadMultiple:Integer; const
      PadChar: AnsiChar) : AnsiString
7317:  Function DecodeBase64( const S, Alphabet : AnsiString; const PadSet : CharSet) : AnsiString
7318:  unit uPSI_cFundamentUtils;
7319:  Const('b64_MIMEBase64','Str').String('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
7320:  Const('b64_UUEncode','String').String('!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_';
7321:  Const('b64_XXEncode','String').String('+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';
7322:  Const('CCHARSET','String').SetString(b64_XXEncode);
7323:  Const('CHEXSET','String').SetString('0123456789ABCDEF
7324:  Const('HEXDIGITS','String').SetString('0123456789ABCDEF
7325:  StHexDigits  : array[0..$F] of Char = '0123456789ABCDEF';
7326:  Const('DIGISET','String').SetString('0123456789
```

```
7327:  Const('LETTERSET','String').SetString('ABCDEFGHIJKLMNOPQRSTUVWXYZ
7328:  Const('DIGISET2','String').SetSet('0123456789
7329:  Const('LETTERSET2','String').SetSet('ABCDEFGHIJKLMNOPQRSTUVWXYZ
7330:  Function CharSetToStr( const C : CharSet) : AnsiString
7331:  Function StrToCharSet( const S : AnsiString) : CharSet
7332:  Function MIMEBase64Decode( const S : AnsiString) : AnsiString
7333:  Function MIMEBase64Encode( const S : AnsiString) : AnsiString
7334:  Function UUDecode( const S : AnsiString) : AnsiString
7335:  Function XXDecode( const S : AnsiString) : AnsiString
7336:  Function BytesToHex( const P : array of Byte; const UpperCase : Boolean) : AnsiString
7337:  Function InterfaceToStrA( const I : IInterface) : AnsiString
7338:  Function InterfaceToStrW( const I : IInterface) : WideString
7339:  Function InterfaceToStr( const I : IInterface) : String
7340:  Function ObjectClassName( const O : TObject) : String
7341:  Function ClassClassName( const C : TClass) : String
7342:  Function ObjectToStr( const O : TObject) : String
7343:  Function ObjectToString( const O : TObject) : String
7344:  Function CharSetToStr( const C : CharSet) : AnsiString
7345:  Function StrToCharSet( const S : AnsiString) : CharSet
7346:  Function HashStrA( const S : AnsiString; const Index : Integer; const Count : Integer; const
      AsciiCaseSensitive : Boolean; const Slots : LongWord) : LongWord
7347:  Function HashStrW(const S:WideString;const Index:Integer;const Count:Integer;const
      AsciiCaseSensitive:Boolean; const Slots:LongWord) : LongWord
7348:  Function HashStr( const S : String; const Index : Integer; const Count : Integer; const
      AsciiCaseSensitive : Boolean; const Slots : LongWord) : LongWord
7349:  Function HashInteger( const I : Integer; const Slots : LongWord) : LongWord
7350:  Function HashLongWord( const I : LongWord; const Slots : LongWord) : LongWord
7351:  Const('Bytes1KB','LongInt').SetInt( 1024);
7352:   SIRegister_IInterface(CL);
7353:  Procedure SelfTestCFundamentUtils
7354:
7355: Function CreateSchedule : IJclSchedule
7356: Function NullStamp : TTimeStamp
7357: Function CompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Int64
7358: Function EqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp) : Boolean
7359: Function IsNullTimeStamp( const Stamp : TTimeStamp) : Boolean
7360:
7361: procedure SIRegister_uwinplot(CL: TPSPascalCompiler);
7362: begin
7363:  AddTypeS('TFunc', 'function(X : Float) : Float;
7364: Function InitGraphics( Width, Height : Integer) : Boolean
7365: Procedure SetWindow( Canvas : TCanvas; X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
7366: Procedure SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
7367: Procedure SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
7368: Procedure GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float)
7369: Procedure GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float)
7370: Procedure SetGraphTitle( Title : String)
7371: Procedure SetOxTitle( Title : String)
7372: Procedure SetOyTitle( Title : String)
7373: Function GetGraphTitle : String
7374: Function GetOxTitle : String
7375: Function GetOyTitle : String
7376: Procedure PlotOxAxis( Canvas : TCanvas)
7377: Procedure PlotOyAxis( Canvas : TCanvas)
7378: Procedure PlotGrid( Canvas : TCanvas; Grid : TGrid)
7379: Procedure WriteGraphTitle( Canvas : TCanvas)
7380: Function SetMaxCurv( NCurv : Byte) : Boolean
7381: Procedure SetPointParam( CurvIndex, Symbol, Size : Integer; Color : TColor)
7382: Procedure SetLineParam( CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor)
7383: Procedure SetCurvLegend( CurvIndex : Integer; Legend : String)
7384: Procedure SetCurvStep( CurvIndex, Step : Integer)
7385: Function GetMaxCurv : Byte
7386: Procedure GetPointParam( CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor)
7387: Procedure GetLineParam(CurvIndex:Integer;var Style:TPenStyle;var Width:Integer; var Color:TColor);
7388: Function GetCurvLegend( CurvIndex : Integer) : String
7389: Function GetCurvStep( CurvIndex : Integer) : Integer
7390: Procedure PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Integer)
7391: Procedure PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Integer)
7392: Procedure PlotCurveWithErrorBars(Canvas : TCanvas; X,Y,S: TVector; Ns,Lb,Ub,CurvIndex:Integer)
7393: Procedure PlotFunc(Canvas: TCanvas; Func: TFunc; Xmin,Xmax: Float; Npt,CurvIndex: Integer)
7394: Procedure WriteLegend( Canvas : TCanvas; NCurv : Integer; ShowPoints, ShowLines : Boolean)
7395: Procedure ConRec( Canvas : TCanvas; Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
7396: Function Xpixel( X : Float) : Integer
7397: Function Ypixel( Y : Float) : Integer
7398: Function Xuser( X : Integer) : Float
7399: Function Yuser( Y : Integer) : Float
7400: end;
7401:
7402: Procedure FFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7403: Procedure IFFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7404: Procedure FFT_Integer( NumSamples : Integer; RealIn, ImagIn : TIntVector; OutArray : TCompVector)
7405: Procedure FFT_Integer_Cleanup
7406: Procedure CalcFrequency(NumSamples,FrequencyIndex: Integer;InArray: TCompVector;var FT : Complex)
7407: //unit uPSI_JclStreams;
7408: Function StreamSeek( Stream : TStream; const Offset : Int64; const Origin : TSeekOrigin) : Int64
7409: Function StreamCopy( Source : TStream; Dest : TStream; BufferSize : Integer) : Int64
7410: Function CompareStreams( A, B : TStream; BufferSize : Integer) : Boolean
7411: Function JCompareFiles( const FileA, FileB : TFileName; BufferSize : Integer) : Boolean
7412:
```

```
7413: procedure SIRegister_FmxUtils(CL: TPSPascalCompiler);
7414: begin
7415:  FindClass('TOBJECT'),'EInvalidDest
7416:  FindClass('TOBJECT'),'EFCantMove
7417:  Procedure fmxCopyFile( const FileName, DestName : string)
7418:  Procedure fmxMoveFile( const FileName, DestName : string)
7419:  Function fmxGetFileSize( const FileName : string) : LongInt
7420:  Function fmxFileDateTime( const FileName : string) : TDateTime
7421:  Function fmxHasAttr( const FileName : string; Attr : Word) : Boolean
7422:  Function fmxExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer):THandle;
7423: end;
7424:
7425: procedure SIRegister_FindFileIter(CL: TPSPascalCompiler);
7426: begin
7427:  SIRegister_IFindFileIterator(CL);
7428:  Function CreateFindFile(const Path:string; IncludeAttr:Integer;out iffi:IFindFileIterator):Bool;
7429: end;
7430:
7431: procedure SIRegister_PCharUtils(CL: TPSPascalCompiler);
7432: begin
7433:  Function SkipWhite( cp : PChar) : PChar
7434:  Function ReadStringDoubleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7435:  Function ReadStringSingleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7436:  Function ReadIdent( cp : PChar; var ident : string) : PChar
7437: end;
7438:
7439: procedure SIRegister_JclStrHashMap(CL: TPSPascalCompiler);
7440: begin
7441:   SIRegister_TStringHashMapTraits(CL);
7442:  Function CaseSensitiveTraits : TStringHashMapTraits
7443:  Function CaseInsensitiveTraits : TStringHashMapTraits
7444:   THashNode', 'record Str : string; Ptr : Pointer; Left : PHashNod'
7445:    +'e; Right : PHashNode; end
7446:   //PHashArray', '^THashArray // will not work
7447:   SIRegister_TStringHashMap(CL);
7448:  THashValue', 'Cardinal
7449:  Function StrHash( const s : string) : THashValue
7450:  Function TextHash( const s : string) : THashValue
7451:  Function DataHash( var AValue, ASize : Cardinal) : THashValue
7452:  Function Iterate_FreeObjects( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7453:  Function Iterate_Dispose( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7454:  Function Iterate_FreeMem( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7455:   SIRegister_TCaseSensitiveTraits(CL);
7456:   SIRegister_TCaseInsensitiveTraits(CL);
7457:
7458:
7459: //************************************************************************unit uPSI_umath;
7460:  Function uExpo( X : Float) : Float
7461:  Function uExp2( X : Float) : Float
7462:  Function uExp10( X : Float) : Float
7463:  Function uLog( X : Float) : Float
7464:  Function uLog2( X : Float) : Float
7465:  Function uLog10( X : Float) : Float
7466:  Function uLogA( X, A : Float) : Float
7467:  Function uIntPower( X : Float; N : Integer): Float
7468:  Function uPower( X, Y : Float) : Float
7469:  Function SgnGamma( X : Float) : Integer
7470:  Function Stirling( X : Float) : Float
7471:  Function StirLog( X : Float) : Float
7472:  Function Gamma( X : Float) : Float
7473:  Function LnGamma( X : Float) : Float
7474: Function DiGamma( X : Float) : Float
7475:  Function TriGamma( X : Float) : Float
7476:  Function IGamma( X : Float) : Float
7477:  Function JGamma( X : Float) : Float
7478:  Function InvGamma( X : Float) : Float
7479:  Function Erf( X : Float) : Float
7480:  Function Erfc( X : Float) : Float
7481: Function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7482: { Correlation coefficient between samples X and Y }
7483: function DBeta(A, B, X : Float) : Float;
7484: { Density of Beta distribution with parameters A and B }
7485: Function LambertW( X : Float; UBranch, Offset : Boolean) : Float
7486: Function Beta(X, Y : Float) : Float
7487: Function Binomial( N, K : Integer) : Float
7488: Function PBinom( N : Integer; P : Float; K : Integer) : Float
7489: Procedure Cholesky( A, L : TMatrix; Lb, Ub : Integer)
7490: Procedure LU_Decomp( A : TMatrix; Lb, Ub : Integer)
7491: Procedure LU_Solve( A : TMatrix; B : TVector; Lb, Ub : Integer; X : TVector)
7492: Function DNorm( X : Float) : Float
7493:
7494: function DGamma(A, B, X : Float) : Float;
7495: { Density of Gamma distribution with parameters A and B }
7496: function DKhi2(Nu : Integer; X : Float) : Float;
7497: { Density of Khi-2 distribution with Nu d.o.f. }
7498: function DStudent(Nu : Integer; X : Float) : Float;
7499: { Density of Student distribution with Nu d.o.f. }
7500: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float;
7501: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
```

```
7502: function IBeta(A, B, X : Float) : Float;
7503: { Incomplete Beta function}
7504: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7505:
7506: procedure SIRegister_unlfit(CL: TPSPascalCompiler);
7507: begin
7508:  Procedure SetOptAlgo( Algo : TOptAlgo)
7509: procedure SetOptAlgo(Algo : TOptAlgo);
7510: { ----------------------------------------------------------------
7511:   Sets the optimization algorithm according to Algo, which must be
7512:   NL_MARQ, NL_SIMP, NL_BFGS, NL_SA, NL_GA. Default is NL_MARQ   }
7513:
7514:  Function GetOptAlgo : TOptAlgo
7515:  Procedure SetMaxParam( N : Byte)
7516:  Function GetMaxParam : Byte
7517:  Procedure SetParamBounds( I : Byte; ParamMin, ParamMax : Float)
7518:  Procedure GetParamBounds( I : Byte; var ParamMin, ParamMax : Float)
7519:  Function NullParam( B : TVector; Lb, Ub : Integer) : Boolean
7520:  Procedure NLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y : TVector; Lb, Ub : Integer; MaxIter :
      Integer; Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7521:  Procedure WNLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y, S : TVector; Lb, Ub:Integer;
      MaxIter:Integer;Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7522:  Procedure SetMCFile( FileName : String)
7523:  Procedure SimFit(RegFunc:TRegFunc;X,Y:TVector;Lb,Ub:Integer;B:TVector;FirstPar,LastPar:Integer;V:TMatrix;
7524:  Procedure WSimFit(RegFunc:TRegFunc; X,Y,S:TVector;Lb,Ub:Integer;B:TVector;FirstPar,
      LastPar:Integer;V:TMatrix);
7525: end;
7526:
7527: (*----------------------------------------------------------------------------*)
7528: procedure SIRegister_usimplex(CL: TPSPascalCompiler);
7529: begin
7530:  Procedure SaveSimplex( FileName : string)
7531:  Procedure Simplex(Func:TFuncNVar; X:TVector;Lb,Ub:Integer; MaxIter:Integer;Tol:Float; var F_min:Float);
7532: end;
7533: (*----------------------------------------------------------------------------*)
7534: procedure SIRegister_uregtest(CL: TPSPascalCompiler);
7535: begin
7536:  Procedure RegTest(Y,Ycalc: TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest)
7537:  Procedure WRegTest(Y,Ycalc,S:TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest);
7538: end;
7539:
7540: procedure SIRegister_ustrings(CL: TPSPascalCompiler);
7541: begin
7542:  Function LTrim( S : String) : String
7543:  Function RTrim( S : String) : String
7544:  Function uTrim( S : String) : String
7545:  Function StrChar( N : Byte; C : Char) : String
7546:  Function RFill( S : String; L : Byte) : String
7547:  Function LFill( S : String; L : Byte) : String
7548:  Function CFill( S : String; L : Byte) : String
7549:  Function Replace( S : String; C1, C2 : Char) : String
7550:  Function Extract( S : String; var Index : Byte; Delim : Char) : String
7551:  Procedure Parse( S : String; Delim : Char; Field : TStrVector; var N : Byte)
7552:  Procedure SetFormat( NumLength, MaxDec : Integer; FloatPoint, NSZero : Boolean)
7553:  Function FloatStr( X : Float) : String
7554:  Function IntStr( N : LongInt) : String
7555:  Function uCompStr( Z : Complex) : String
7556: end;
7557:
7558: procedure SIRegister_uhyper(CL: TPSPascalCompiler);
7559: begin
7560:  Function uSinh( X : Float) : Float
7561:  Function uCosh( X : Float) : Float
7562:  Function uTanh( X : Float) : Float
7563:  Function uArcSinh( X : Float) : Float
7564:  Function uArcCosh( X : Float) : Float
7565:  Function ArcTanh( X : Float) : Float
7566:  Procedure SinhCosh( X : Float; var SinhX, CoshX : Float)
7567: end;
7568:
7569: procedure SIRegister_urandom(CL: TPSPascalCompiler);
7570: begin
7571: type RNG_Type =
7572:   (RNG_MWC,      { Multiply-With-Carry }
7573:    RNG_MT,       { Mersenne Twister }
7574:    RNG_UVAG);    { Universal Virtual Array Generator }
7575:  Procedure SetRNG( RNG : RNG_Type)
7576:  Procedure InitGen( Seed : RNG_IntType)
7577:  Procedure SRand( Seed : RNG_IntType)
7578:  Function IRanGen : RNG_IntType
7579:  Function IRanGen31 : RNG_IntType
7580:  Function RanGen1 : Float
7581:  Function RanGen2 : Float
7582:  Function RanGen3 : Float
7583:  Function RanGen53 : Float
7584: end;
7585:
7586: //  Optimization by Simulated Annealing
7587: procedure SIRegister_usimann(CL: TPSPascalCompiler);
```

```
7588: begin
7589:  Procedure InitSAParams( NT, NS, NCycles : Integer; RT : Float)
7590:  Procedure SA_CreateLogFile( FileName : String)
7591:  Procedure SimAnn(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7592: end;
7593:
7594: procedure SIRegister_uranuvag(CL: TPSPascalCompiler);
7595: begin
7596:  Procedure InitUVAGbyString( KeyPhrase : string)
7597:  Procedure InitUVAG( Seed : RNG_IntType)
7598:  Function IRanUVAG : RNG_IntType
7599: end;
7600:
7601: procedure SIRegister_ugenalg(CL: TPSPascalCompiler);
7602: begin
7603:  Procedure InitGAParams( NP, NG : Integer; SR, MR, HR : Float)
7604:  Procedure GA_CreateLogFile( LogFileName : String)
7605:  Procedure GenAlg(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7606: end;
7607:
7608:  TVector', 'array of Float
7609: procedure SIRegister_uqsort(CL: TPSPascalCompiler);
7610: begin
7611:  Procedure QSort( X : TVector; Lb, Ub : Integer)
7612:  Procedure DQSort( X : TVector; Lb, Ub : Integer)
7613: end;
7614:
7615: procedure SIRegister_uinterv(CL: TPSPascalCompiler);
7616: begin
7617:  Procedure Interval( X1, X2 : Float; MinDiv, MaxDiv : Integer; var Min, Max, Step : Float)
7618:  Procedure AutoScale(X: TVector; Lb, Ub : Integer; Scale : TScale; var XMin, XMax, XStep:Float)
7619: end;
7620:
7621: procedure SIRegister_D2XXUnit(CL: TPSPascalCompiler);
7622: begin
7623:   FT_Result', 'Integer
7624:   //TDWordptr', '^DWord // will not work
7625:   TFT_Program_Data', 'record Signature1 : DWord; Signature2 : DWor'
7626:    d; Version : DWord; VendorID : Word; ProductID : Word; Manufacturer : PCha'
7627:    r; ManufacturerID : PChar; Description : PChar; SerialNumber : PChar; MaxP'
7628:    ower : Word; PnP : Word; SelfPowered : Word; RemoteWakeup : Word; Rev4 : B'
7629:    yte; IsoIn : Byte; IsoOut : Byte; PullDownEnable : Byte; SerNumEnable : By'
7630:    te; USBVersionEnable : Byte; USBVersion : Word; Rev5 : Byte; IsoInA : Byte'
7631:    ; IsoInB : Byte; IsoOutA : Byte; IsoOutB : Byte; PullDownEnable5 : Byte; S'
7632:    erNumEnable5 : Byte; USBVersionEnable5 : Byte; USBVersion5 : Word; AIsHigh'
7633:    Current : Byte; BIsHighCurrent : Byte; IFAIsFifo : Byte; IFAIsFifoTar : By'
7634:    te; IFAIsFastSer : Byte; AIsVCP : Byte; IFBIsFifo : Byte; IFBIsFifoTar : B'
7635:    yte; IFBIsFastSer : Byte; BIsVCP : Byte; UseExtOsc : Byte; HighDriveIOs : '
7636:    Byte; EndpointSize : Byte; PullDownEnableR : Byte; SerNumEnableR : Byte; I'
7637:    nvertTXD : Byte; InvertRXD : Byte; InvertRTS : Byte; InvertCTS : Byte; Inv'
7638:    ertDTR : Byte; InvertDSR : Byte; InvertDCD : Byte; InvertRI : Byte; Cbus0 '
7639:    : Byte; Cbus1 : Byte; Cbus2 : Byte; Cbus3 : Byte; Cbus4 : Byte; RIsVCP : B'
7640:    yte; end
7641: end;
7642:
7643:
7644: //************************************ PaintFX*************************
7645: procedure SIRegister_TJvPaintFX(CL: TPSPascalCompiler);
7646: begin
7647:   //with RegClassS(CL,'TComponent', 'TJvPaintFX') do
7648:   with AddClassN(FindClass('TComponent'),'TJvPaintFX') do begin
7649:     Procedure Solarize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7650:     Procedure Posterize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7651:     Procedure Blend( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7652:     Procedure Blend2( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7653:     Procedure ExtractColor( const Dst : TBitmap; AColor : TColor)
7654:     Procedure ExcludeColor( const Dst : TBitmap; AColor : TColor)
7655:     Procedure Turn( Src, Dst : TBitmap)
7656:     Procedure TurnRight( Src, Dst : TBitmap)
7657:     Procedure HeightMap( const Dst : TBitmap; Amount : Integer)
7658:     Procedure TexturizeTile( const Dst : TBitmap; Amount : Integer)
7659:     Procedure TexturizeOverlap( const Dst : TBitmap; Amount : Integer)
7660:     Procedure RippleRandom( const Dst : TBitmap; Amount : Integer)
7661:     Procedure RippleTooth( const Dst : TBitmap; Amount : Integer)
7662:     Procedure RippleTriangle( const Dst : TBitmap; Amount : Integer)
7663:     Procedure Triangles( const Dst : TBitmap; Amount : Integer)
7664:     Procedure DrawMandelJulia(const Dst: TBitmap; x0,y0,x1,y1:Single; Niter:Integer; Mandel:Boolean)
7665:     Procedure FilterXBlue( const Dst : TBitmap; Min, Max : Integer)
7666:     Procedure FilterXGreen( const Dst : TBitmap; Min, Max : Integer)
7667:     Procedure FilterXRed( const Dst : TBitmap; Min, Max : Integer)
7668:     Procedure FilterBlue( const Dst : TBitmap; Min, Max : Integer)
7669:     Procedure FilterGreen( const Dst : TBitmap; Min, Max : Integer)
7670:     Procedure FilterRed( const Dst : TBitmap; Min, Max : Integer)
7671:     Procedure Emboss( var Bmp : TBitmap)
7672:     Procedure Plasma( Src1, Src2, Dst : TBitmap; Scale, Turbulence : Single)
7673:     Procedure Shake( Src, Dst : TBitmap; Factor : Single)
7674:     Procedure ShakeDown( Src, Dst : TBitmap; Factor : Single)
7675:     Procedure KeepBlue( const Dst : TBitmap; Factor : Single)
7676:     Procedure KeepGreen( const Dst : TBitmap; Factor : Single)
```

```
7677:     Procedure KeepRed( const Dst : TBitmap; Factor : Single)
7678:     Procedure Mandelbrot( const Dst : TBitmap; Factor : Integer)
7679:     Procedure MaskMandelbrot( const Dst : TBitmap; Factor : Integer)
7680:     Procedure FoldRight( Src1, Src2, Dst : TBitmap; Amount : Single)
7681:     Procedure QuartoOpaque( Src, Dst : TBitmap)
7682:     Procedure SemiOpaque( Src, Dst : TBitmap)
7683:     Procedure ShadowDownLeft( const Dst : TBitmap)
7684:     Procedure ShadowDownRight( const Dst : TBitmap)
7685:     Procedure ShadowUpLeft( const Dst : TBitmap)
7686:     Procedure ShadowUpRight( const Dst : TBitmap)
7687:     Procedure Darkness( const Dst : TBitmap; Amount : Integer)
7688:     Procedure Trace( const Dst : TBitmap; Intensity : Integer)
7689:     Procedure FlipRight( const Dst : TBitmap)
7690:     Procedure FlipDown( const Dst : TBitmap)
7691:     Procedure SpotLight( const Dst : TBitmap; Amount : Integer; Spot : TRect)
7692:     Procedure SplitLight( const Dst : TBitmap; Amount : Integer)
7693:     Procedure MakeSeamlessClip( var Dst : TBitmap; Seam : Integer)
7694:     Procedure Wave( const Dst : TBitmap; Amount, Inference, Style : Integer)
7695:     Procedure Mosaic( const Bm : TBitmap; Size : Integer)
7696:     Procedure SmoothRotate( var Src, Dst : TBitmap; CX, CY : Integer; Angle : Single)
7697:     Procedure SmoothResize( var Src, Dst : TBitmap)
7698:     Procedure Twist( var Bmp, Dst : TBitmap; Amount : Integer)
7699:     Procedure SplitBlur( const Dst : TBitmap; Amount : Integer)
7700:     Procedure GaussianBlur( const Dst : TBitmap; Amount : Integer)
7701:     Procedure Smooth( const Dst : TBitmap; Weight : Integer)
7702:     Procedure GrayScale( const Dst : TBitmap)
7703:     Procedure AddColorNoise( const Dst : TBitmap; Amount : Integer)
7704:     Procedure AddMonoNoise( const Dst : TBitmap; Amount : Integer)
7705:     Procedure Contrast( const Dst : TBitmap; Amount : Integer)
7706:     Procedure Lightness( const Dst : TBitmap; Amount : Integer)
7707:     Procedure Saturation( const Dst : TBitmap; Amount : Integer)
7708:     Procedure Spray( const Dst : TBitmap; Amount : Integer)
7709:     Procedure AntiAlias( const Dst : TBitmap)
7710:     Procedure AntiAliasRect( const Dst : TBitmap; XOrigin, YOrigin, XFinal, YFinal : Integer)
7711:     Procedure SmoothPoint( const Dst : TBitmap; XK, YK : Integer)
7712:     Procedure FishEye( var Bmp, Dst : TBitmap; Amount : Single)
7713:     Procedure Marble( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7714:     Procedure Marble2( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7715:     Procedure Marble3( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7716:     Procedure Marble4( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7717:     Procedure Marble5( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7718:     Procedure Marble6( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7719:     Procedure Marble7( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7720:     Procedure Marble8( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence: Integer)
7721:     Procedure SqueezeHor( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7722:     Procedure SplitRound( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7723:     Procedure Tile( Src, Dst : TBitmap; Amount : Integer)
7724:     Procedure Stretch( Src, Dst : TBitmap; Filter : TFilterProc; AWidth : Single)
7725:     Procedure Grow( Src1, Src2, Dst : TBitmap; Amount : Single; X, Y : Integer)
7726:     Procedure Invert( Src : TBitmap)
7727:     Procedure MirrorRight( Src : TBitmap)
7728:     Procedure MirrorDown( Src : TBitmap)
7729:   end;
7730: end;
7731:
7732: (*----------------------------------------------------------------------------*)
7733: procedure SIRegister_JvPaintFX(CL: TPSPascalCompiler);
7734: begin
7735:   AddTypeS('TLightBrush', '( lbBrightness, lbContrast, lbSaturation, lbFishe'
7736:    +'ye, lbrotate, lbtwist, lbrimple, mbHor, mbTop, mbBottom, mbDiamond, mbWast'
7737:    +'e, mbRound, mbRound2, mbSplitRound, mbSplitWaste )
7738:   SIRegister_TJvPaintFX(CL);
7739:  Function SplineFilter( Value : Single) : Single
7740:  Function BellFilter( Value : Single) : Single
7741:  Function TriangleFilter( Value : Single) : Single
7742:  Function BoxFilter( Value : Single) : Single
7743:  Function HermiteFilter( Value : Single) : Single
7744:  Function Lanczos3Filter( Value : Single) : Single
7745:  Function MitchellFilter( Value : Single) : Single
7746: end;
7747:
7748:
7749: (*----------------------------------------------------------------------------*)
7750: procedure SIRegister_Chart(CL: TPSPascalCompiler);
7751: begin
7752:  'TeeMsg_DefaultFunctionName','String').SetString( 'TeeFunction
7753:  TeeMsg_DefaultSeriesName','String').SetString( 'Series
7754:  TeeMsg_DefaultToolName','String').SetString( 'ChartTool
7755:  ChartComponentPalette','String').SetString( 'TeeChart
7756:  TeeMaxLegendColumns',LongInt').SetInt( 2);
7757:  TeeDefaultLegendSymbolWidth','LongInt').SetInt( 20);
7758:  TeeTitleFootDistance,LongInt).SetInt( 5);
7759:   SIRegister_TCustomChartWall(CL);
7760:   SIRegister_TChartWall(CL);
7761:   SIRegister_TChartLegendGradient(CL);
7762:   TLegendStyle', '( lsAuto, lsSeries, lsValues, lsLastValues, lsSeriesGroups )
7763:   TLegendAlignment', '( laLeft, laRight, laTop, laBottom )
7764:   FindClass('TOBJECT'),'LegendException
7765:   TOnGetLegendText', 'Procedure ( Sender : TCustomAxisPanel; Legen'
```

```
7766:    +'dStyle : TLegendStyle; Index : Integer; var LegendText : String)
7767:   FindClass('TOBJECT'),'TCustomChartLegend
7768:   TLegendSymbolSize', '( lcsPercent, lcsPixels )
7769:   TLegendSymbolPosition', '( spLeft, spRight )
7770:   TSymbolDrawEvent','Procedure(Sender:TObject;Series:TChartSeries;ValueIndex:Integer;R:TRect);
7771:   TSymbolCalcHeight', 'Function  : Integer
7772:   SIRegister_TLegendSymbol(CL);
7773:   SIRegister_TTeeCustomShapePosition(CL);
7774:   TCheckBoxesStyle', '( cbsCheck, cbsRadio )
7775:   SIRegister_TLegendTitle(CL);
7776:   SIRegister_TLegendItem(CL);
7777:   SIRegister_TLegendItems(CL);
7778:   TLegendCalcSize', 'Procedure ( Sender : TCustomChartLegend; var ASize : Integer)
7779:   FindClass('TOBJECT'),'TCustomChart
7780:   SIRegister_TCustomChartLegend(CL);
7781:   SIRegister_TChartLegend(CL);
7782:   SIRegister_TChartTitle(CL);
7783:   SIRegister_TChartFootTitle(CL);
7784:   TChartClick', 'Procedure ( Sender : TCustomChart; Button : TMous'
7785:    +'eButton; Shift : TShiftState; X, Y : Integer)
7786:   TChartClickAxis', 'Procedure ( Sender : TCustomChart; Axis : TCh'
7787:    +'artAxis; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7788:   TChartClickSeries', 'Procedure ( Sender : TCustomChart; Series :'
7789:    +' TChartSeries; ValueIndex : Integer; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7790:   TChartClickTitle', 'Procedure ( Sender : TCustomChart; ATitle : '
7791:    +'TChartTitle; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7792:   TOnGetLegendPos', 'Procedure ( Sender : TCustomChart; Index : In'
7793:    +'teger; var X, Y, XColor : Integer)
7794:   TOnGetLegendRect', 'Procedure ( Sender : TCustomChart; var Rect : TRect)
7795:   TAxisSavedScales', 'record Auto : Boolean; AutoMin : Boolean; Au'
7796:    +'toMax : Boolean; Min : Double; Max : Double; end
7797:   TAllAxisSavedScales', 'array of TAxisSavedScales
7798:   SIRegister_TChartBackWall(CL);
7799:   SIRegister_TChartRightWall(CL);
7800:   SIRegister_TChartBottomWall(CL);
7801:   SIRegister_TChartLeftWall(CL);
7802:   SIRegister_TChartWalls(CL);
7803:   TChartAllowScrollEvent','Procedure(Sender:TChartAxis;var AMin,AMax:Double;var AllowScroll:Boolean);
7804:   SIRegister_TCustomChart(CL);
7805:   SIRegister_TChart(CL);
7806:   SIRegister_TTeeSeriesTypes(CL);
7807:   SIRegister_TTeeToolTypes(CL);
7808:   SIRegister_TTeeDragObject(CL);
7809:   SIRegister_TColorPalettes(CL);
7810:  Procedure RegisterTeeSeries(ASeriesClass:TChartSeriesClass;ADescription,
      AGalleryPage:PString;ANumGallerySeries:Integer;
7811:  Procedure RegisterTeeSeries1( ASeriesClass : TChartSeriesClass; ADescription : PString);
7812:  Procedure RegisterTeeFunction(AFunctClass:TTeeFunctionClass;ADescription,
      AGalleryPage:PString;ANumGallerySeries: Int;
7813:  Procedure RegisterTeeBasicFunction( AFunctionClass : TTeeFunctionClass; ADescription : PString)
7814:  Procedure RegisterTeeSeriesFunction(ASeriesClass: TChartSeriesClass;AFunctionClass:TTeeFunctionClass;
      ADescription, AGalleryPage : PString; ANumGallerySeries : Integer; ASubIndex : Integer)
7815:  Procedure UnRegisterTeeSeries( const ASeriesList : array of TChartSeriesClass)
7816:  Procedure UnRegisterTeeFunctions( const AFunctionList : array of TTeeFunctionClass)
7817:  Procedure AssignSeries( var OldSeries, NewSeries : TChartSeries)
7818:  Function CreateNewTeeFunction( ASeries : TChartSeries; AClass : TTeeFunctionClass) : TTeeFunction
7819:  Function CreateNewSeries( AOwner : TComponent; AChart : TCustomAxisPanel; AClass : TChartSeriesClass;
      AFunctionClass : TTeeFunctionClass) : TChartSeries
7820:  Function CloneChartSeries( ASeries : TChartSeries) : TChartSeries;
7821:  Function CloneChartSeries1( ASeries : TChartSeries; AChart : TCustomAxisPanel) : TChartSeries;
7822:  Function CloneChartSeries2(ASeries:TChartSeries;AOwner:TComponent;AChart:TCustomAxisPanel):TChartSeries;;
7823:  Function CloneChartTool( ATool : TTeeCustomTool; AOwner : TComponent) : TTeeCustomTool
7824:  Function ChangeSeriesType( var ASeries : TChartSeries; NewType : TChartSeriesClass) : TChartSeries
7825:  Procedure ChangeAllSeriesType( AChart : TCustomChart; AClass : TChartSeriesClass)
7826:  Function GetNewSeriesName( AOwner : TComponent) : TComponentName
7827:  Procedure RegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7828:  Procedure UnRegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7829:  Function GetGallerySeriesName( ASeries : TChartSeries) : String
7830:  Procedure PaintSeriesLegend(ASeries:TChartSeries; ACanvas:TCanvas; const R:TRect;ReferenceChart:
      TCustomChart);
7831:   SIRegister_TChartTheme(CL);
7832:   //TChartThemeClass', 'class of TChartTheme
7833:   //TCanvasClass', 'class of TCanvas3D
7834:  Function SeriesNameOrIndex( ASeries : TCustomChartSeries) : String
7835:  Function SeriesTitleOrName( ASeries : TCustomChartSeries) : String
7836:  Procedure FillSeriesItems( AItems : TStrings; AList : TCustomSeriesList; UseTitles : Boolean)
7837:  Procedure ShowMessageUser( const S : String)
7838:  Function HasNoMandatoryValues( ASeries : TChartSeries) : Boolean
7839:  Function HasLabels( ASeries : TChartSeries) : Boolean
7840:  Function HasColors( ASeries : TChartSeries) : Boolean
7841:  Function SeriesGuessContents( ASeries : TChartSeries) : TeeFormatFlag
7842:  Procedure TeeDrawBitmapEditor( Canvas : TCanvas; Element : TCustomChartElement; Left, Top : Integer)
7843: end;
7844:
7845:
7846: procedure SIRegister_TeeProcs(CL: TPSPascalCompiler);
7847: begin
7848:  //'TeeFormBorderStyle','').SetString( bsNone);
7849:   SIRegister_TMetafile(CL);
```

```
7850:  'TeeDefVerticalMargin','LongInt').SetInt( 4);
7851:  'TeeDefHorizMargin','LongInt').SetInt( 3);
7852:  'crTeeHand','LongInt').SetInt( TCursor ( 2020 ));
7853:  'TeeMsg_TeeHand','String').SetString( 'crTeeHand
7854:  'TeeNormalPrintDetail','LongInt').SetInt( 0);
7855:  'TeeHighPrintDetail','LongInt').SetInt( - 100);
7856:  'TeeDefault_PrintMargin','LongInt').SetInt( 15);
7857:  'MaxDefaultColors','LongInt').SetInt( 19);
7858:  'TeeTabDelimiter','Char').SetString( #9);
7859:   TDateTimeStep', '( dtOneMicroSecond, dtOneMillisecond, dtOneSeco'
7860:    +'nd, dtFiveSeconds, dtTenSeconds, dtFifteenSeconds, dtThirtySeconds, dtOneM'
7861:    +'inute, dtFiveMinutes, dtTenMinutes, dtFifteenMinutes, dtThirtyMinutes, dtO'
7862:    +'neHour, dtTwoHours, dtSixHours, dtTwelveHours, dtOneDay, dtTwoDays, dtThre'
7863:    +'eDays, dtOneWeek, dtHalfMonth, dtOneMonth, dtTwoMonths, dtThreeMonths, dtF'
7864:    +'ourMonths, dtSixMonths, dtOneYear, dtNone )
7865:   SIRegister_TCustomPanelNoCaption(CL);
7866:   FindClass('TOBJECT'),'TCustomTeePanel
7867:   SIRegister_TZoomPanning(CL);
7868:   SIRegister_TTeeEvent(CL);
7869:   //SIRegister_TTeeEventListeners(CL);
7870:   TTeeMouseEventKind', '( meDown, meUp, meMove )
7871:   SIRegister_TTeeMouseEvent(CL);
7872:   SIRegister_TCustomTeePanel(CL);
7873:   //TChartGradient', 'TTeeGradient
7874:   //TChartGradientClass', 'class of TChartGradient
7875:   TPanningMode', '( pmNone, pmHorizontal, pmVertical, pmBoth )
7876:   SIRegister_TTeeZoomPen(CL);
7877:   SIRegister_TTeeZoomBrush(CL);
7878:   TTeeZoomDirection', '( tzdHorizontal, tzdVertical, tzdBoth )
7879:   SIRegister_TTeeZoom(CL);
7880:   FindClass('TOBJECT'),'TCustomTeePanelExtended
7881:   TTeeBackImageMode', '( pbmStretch, pbmTile, pbmCenter, pbmCustom )
7882:   SIRegister_TBackImage(CL);
7883:   SIRegister_TCustomTeePanelExtended(CL);
7884:   //TChartBrushClass', 'class of TChartBrush
7885:   SIRegister_TTeeCustomShapeBrushPen(CL);
7886:   TChartObjectShapeStyle', '( fosRectangle, fosRoundRectangle, fosEllipse )
7887:   TTextFormat', '( ttfNormal, ttfHtml )
7888:   SIRegister_TTeeCustomShape(CL);
7889:   SIRegister_TTeeShape(CL);
7890:   SIRegister_TTeeExportData(CL);
7891:  Function TeeStr( const Num : Integer) : String
7892:  Function DateTimeDefaultFormat( const AStep : Double) : String
7893:  Function TEEDaysInMonth( Year, Month : Word) : Word
7894:  Function FindDateTimeStep( const StepValue : Double) : TDateTimeStep
7895:  Function NextDateTimeStep( const AStep : Double) : Double
7896:  Function PointInLine( const P : TPoint; const px, py, qx, qy : Integer) : Boolean;
7897:  Function PointInLine1( const P, FromPoint, ToPoint : TPoint) : Boolean;
7898:  Function PointInLine2(const P,FromPoint,ToPoint:TPoint;const TolerancePixels:Integer):Boolean;
7899:  Function PointInLine3( const P : TPoint; const px, py, qx, qy, TolerancePixels:Integer):Boolean;
7900:  Function PointInLineTolerance(const P:TPoint;const px,py,qx,qy,TolerancePixels:Integer):Boolean;
7901:  Function PointInPolygon( const P : TPoint; const Poly : array of TPoint) : Boolean
7902:  Function PointInTriangle( const P, P0, P1, P2 : TPoint) : Boolean;
7903:  Function PointInTriangle1( const P : TPoint; X0, X1, Y0, Y1 : Integer) : Boolean;
7904:  Function PointInHorizTriangle( const P : TPoint; Y0, Y1, X0, X1 : Integer) : Boolean
7905:  Function PointInEllipse( const P : TPoint; const Rect : TRect) : Boolean;
7906:  Function PointInEllipse1( const P: TPoint;Left,Top,Right, Bottom : Integer) : Boolean;
7907:  Function DelphiToLocalFormat( const Format : String) : String
7908:  Function LocalToDelphiFormat( const Format : String) : String
7909:  Procedure TEEEnableControls(Enable: Boolean; const ControlArray : array of TControl)
7910:  Function TeeRoundDate(const ADate : TDateTime; AStep : TDateTimeStep) : TDateTime
7911:  Procedure TeeDateTimeIncrement(IsDateTime:Boolean;Increment:Boolean;var Value:Double;const
      AnIncrement:Double; tmpWhichDateTime:TDateTimeStep)
7912:   TTeeSortCompare', 'Function ( a, b : Integer) : Integer
7913:   TTeeSortSwap', 'Procedure ( a, b : Integer)
7914:  Procedure TeeSort(StartIndex,EndIndex:Integer;CompareFunc:TTeeSortCompare;SwapFunc:TTeeSortSwap);
7915:  Function TeeGetUniqueName( AOwner : TComponent; const AStartName : String) : string
7916:  Function TeeExtractField( St : String; Index : Integer) : String;
7917:  Function TeeExtractField1( St : String; Index : Integer; const Separator : String) : String;
7918:  Function TeeNumFields( St : String) : Integer;
7919:  Function TeeNumFields1( const St, Separator : String) : Integer;
7920:  Procedure TeeGetBitmapEditor( AObject : TObject; var Bitmap : TBitmap)
7921:  Procedure TeeLoadBitmap( Bitmap : TBitmap; const Name1, Name2 : String)
7922:  // TColorArray', 'array of TColor
7923:  Function GetDefaultColor( const Index : Integer) : TColor
7924:  Procedure SetDefaultColorPalette;
7925:  Procedure SetDefaultColorPalette1( const Palette : array of TColor);
7926:  'TeeCheckBoxSize','LongInt').SetInt( 11);
7927:  Procedure TeeDrawCheckBox(x,y:Integer;Canvas:TCanvas;Checked:Boolean;ABackColor:TColor;CheckBox:Boolean);
7928:  Function TEEStrToFloatDef( const S : string; const Default : Extended) : Extended
7929:  Function TryStrToFloat( const S : String; var Value : Double) : Boolean
7930:  Function CrossingLines( const X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Double; out x, y : Double) : Boolean
7931:  Procedure TeeTranslateControl( AControl : TControl);
7932:  Procedure TeeTranslateControl1( AControl : TControl; const ExcludeChilds : array of TControl);
7933:  Function ReplaceChar( const AString : String; const Search : Char; const Replace : Char) : String
7934:  //Procedure RectToFourPoints( const ARect : TRect; const Angle : Double; var P : TFourPoints)
7935:  Function TeeAntiAlias( Panel : TCustomTeePanel; ChartRect : Boolean) : TBitmap
7936:  //Procedure DrawBevel(Canvas:TTeeCanvas;Bevel:TPanelBevel;var R:TRect;Width:Integer;Round:Integer);
7937:  //Function ScreenRatio( ACanvas : TCanvas3D) : Double
```

```
7938:  Function TeeReadBoolOption( const AKey : String; DefaultValue : Boolean) : Boolean
7939:  Procedure TeeSaveBoolOption( const AKey : String; Value : Boolean)
7940:  Function TeeReadIntegerOption( const AKey : String; DefaultValue : Integer) : Integer
7941:  Procedure TeeSaveIntegerOption( const AKey : String; Value : Integer)
7942:  Function TeeReadStringOption( const AKey, DefaultValue : String) : String
7943:  Procedure TeeSaveStringOption( const AKey, Value : String)
7944:  Function TeeDefaultXMLEncoding : String
7945:  Procedure ConvertTextToXML( Stream : TStream; XMLHeader : Boolean)
7946:   TeeWindowHandle', 'Integer
7947:  Procedure TeeGotoURL( Handle : TeeWindowHandle; const URL : String)
7948:  Procedure HtmlTextOut( ACanvas : TCanvas; x, y : Integer; Text : String)
7949:  Function HtmlTextExtent( ACanvas : TCanvas; const Text : String) : TSize
7950: end;
7951:
7952:
7953: using mXBDEUtils
7954: ********************************************************************************
7955:  Procedure SetAlias( aAlias, aDirectory : String)
7956:  Procedure CheckRegistryEntry(Reg:TRegistry;const Path,Value:String;const Default,
      Desired:Variant;Size:Byte);
7957:  Function GetFileVersionNumber( const FileName : String) : TVersionNo
7958:  Procedure SetBDE( aPath, aNode, aValue : String)
7959:  function RestartDialog(Wnd: HWnd; Reason: PChar; Flags: Integer): Integer; stdcall;
7960:  Function GetSystemDirectory( lpBuffer : string; uSize : UINT) : UINT
7961:  Function GetSystemDirectoryW( lpBuffer : pchar; uSize : UINT) : UINT
7962:  Function GetTempPath( nBufferLength : DWORD; lpBuffer : string) : DWORD
7963:  Function GetWindowsDirectoryW( nBufferLength : DWORD; lpBuffer : string) : DWORD
7964:  Function GetTempFileName(lpPathName,lpPrefixString:string;uUnique:UINT;lpTempFileName:string):UINT;
7965:
7966:
7967: procedure SIRegister_cDateTime(CL: TPSPascalCompiler);
7968: begin
7969: AddClassN(FindClass('TOBJECT'),'EDateTime
7970: Function DatePart( const D : TDateTime) : Integer
7971: Function TimePart( const D : TDateTime) : Double
7972: Function Century( const D : TDateTime) : Word
7973: Function Year( const D : TDateTime) : Word
7974: Function Month( const D : TDateTime) : Word
7975: Function Day( const D : TDateTime) : Word
7976: Function Hour( const D : TDateTime) : Word
7977: Function Minute( const D : TDateTime) : Word
7978: Function Second( const D : TDateTime) : Word
7979: Function Millisecond( const D : TDateTime) : Word
7980: ('OneDay','Extended').setExtended( 1.0);
7981: ('OneHour','Extended').SetExtended( OneDay / 24);
7982: ('OneMinute','Extended').SetExtended( OneHour / 60);
7983: ('OneSecond','Extended').SetExtended( OneMinute / 60);
7984: ('OneMillisecond','Extended').SetExtended( OneSecond / 1000);
7985: ('OneWeek','Extended').SetExtended( OneDay * 7);
7986: ('HoursPerDay','Extended').SetExtended( 24);
7987: ('MinutesPerHour','Extended').SetExtended( 60);
7988: ('SecondsPerMinute','Extended').SetExtended( 60);
7989: Procedure SetYear( var D : TDateTime; const Year : Word)
7990: Procedure SetMonth( var D : TDateTime; const Month : Word)
7991: Procedure SetDay( var D : TDateTime; const Day : Word)
7992: Procedure SetHour( var D : TDateTime; const Hour : Word)
7993: Procedure SetMinute( var D : TDateTime; const Minute : Word)
7994: Procedure SetSecond( var D : TDateTime; const Second : Word)
7995: Procedure SetMillisecond( var D : TDateTime; const Milliseconds : Word)
7996: Function IsEqual( const D1, D2 : TDateTime) : Boolean;
7997: Function IsEqual1( const D1 : TDateTime; const Ye, Mo, Da : Word):Boolean;
7998: Function IsEqual2( const D1 : TDateTime; const Ho, Mi, Se, ms : Word):Boolean;
7999: Function IsAM( const D : TDateTime) : Boolean
8000: Function IsPM( const D : TDateTime) : Boolean
8001: Function IsMidnight( const D : TDateTime) : Boolean
8002: Function IsNoon( const D : TDateTime) : Boolean
8003: Function IsSunday( const D : TDateTime) : Boolean
8004: Function IsMonday( const D : TDateTime) : Boolean
8005: Function IsTuesday( const D : TDateTime) : Boolean
8006: Function IsWednesday( const D : TDateTime) : Boolean
8007: Function IsThursday( const D : TDateTime) : Boolean
8008: Function IsFriday( const D : TDateTime) : Boolean
8009: Function IsSaturday( const D : TDateTime) : Boolean
8010: Function IsWeekend( const D : TDateTime) : Boolean
8011: Function Noon( const D : TDateTime) : TDateTime
8012: Function Midnight( const D : TDateTime) : TDateTime
8013: Function FirstDayOfMonth( const D : TDateTime) : TDateTime
8014: Function LastDayOfMonth( const D : TDateTime) : TDateTime
8015: Function NextWorkday( const D : TDateTime) : TDateTime
8016: Function PreviousWorkday( const D : TDateTime) : TDateTime
8017: Function FirstDayOfYear( const D : TDateTime) : TDateTime
8018: Function LastDayOfYear( const D : TDateTime) : TDateTime
8019: Function EasterSunday( const Year : Word) : TDateTime
8020: Function GoodFriday( const Year : Word) : TDateTime
8021: Function AddMilliseconds( const D : TDateTime; const N : Int64) : TDateTime
8022: Function AddSeconds( const D : TDateTime; const N : Int64) : TDateTime
8023: Function AddMinutes( const D : TDateTime; const N : Integer) : TDateTime
8024: Function AddHours( const D : TDateTime; const N : Integer) : TDateTime
8025: Function AddDays( const D : TDateTime; const N : Integer) : TDateTime
```

```
8026: Function AddWeeks( const D : TDateTime; const N : Integer) : TDateTime
8027: Function AddMonths( const D : TDateTime; const N : Integer) : TDateTime
8028: Function AddYears( const D : TDateTime; const N : Integer) : TDateTime
8029: Function DayOfYear( const Ye, Mo, Da : Word) : Integer
8030: Function DayOfYear( const D : TDateTime) : Integer
8031: Function DaysInMonth( const Ye, Mo : Word) : Integer
8032: Function DaysInMonth( const D : TDateTime) : Integer
8033: Function DaysInYear( const Ye : Word) : Integer
8034: Function DaysInYearDate( const D : TDateTime) : Integer
8035: Function WeekNumber( const D : TDateTime) : Integer
8036: Function ISOFirstWeekOfYear( const Ye : Word) : TDateTime
8037: Procedure ISOWeekNumber( const D : TDateTime; var WeekNumber, WeekYear : Word)
8038: Function DiffMilliseconds( const D1, D2 : TDateTime) : Int64
8039: Function DiffSeconds( const D1, D2 : TDateTime) : Integer
8040: Function DiffMinutes( const D1, D2 : TDateTime) : Integer
8041: Function DiffHours( const D1, D2 : TDateTime) : Integer
8042: Function DiffDays( const D1, D2 : TDateTime) : Integer
8043: Function DiffWeeks( const D1, D2 : TDateTime) : Integer
8044: Function DiffMonths( const D1, D2 : TDateTime) : Integer
8045: Function DiffYears( const D1, D2 : TDateTime) : Integer
8046: Function GMTBias : Integer
8047: Function GMTTimeToLocalTime( const D : TDateTime) : TDateTime
8048: Function LocalTimeToGMTTime( const D : TDateTime) : TDateTime
8049: Function NowAsGMTTime : TDateTime
8050: Function DateTimeToISO8601String( const D : TDateTime) : AnsiString
8051: Function ISO8601StringToTime( const D : AnsiString) : TDateTime
8052: Function ISO8601StringAsDateTime( const D : AnsiString) : TDateTime
8053: Function DateTimeToANSI( const D : TDateTime) : Integer
8054: Function ANSIToDateTime( const Julian : Integer) : TDateTime
8055: Function DateTimeToISOInteger( const D : TDateTime) : Integer
8056: Function DateTimeToISOString( const D : TDateTime) : AnsiString
8057: Function ISOIntegerToDateTime( const ISOInteger : Integer) : TDateTime
8058: Function TwoDigitRadix2000YearToYear( const Y : Integer) : Integer
8059: Function DateTimeAsElapsedTime(const D:TDateTime; const IncludeMilliseconds:Boolean):AnsiString
8060: Function UnixTimeToDateTime( const UnixTime : LongWord) : TDateTime
8061: Function DateTimeToUnixTime( const D : TDateTime) : LongWord
8062: Function EnglishShortDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8063: Function EnglishShortDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8064: Function EnglishLongDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8065: Function EnglishLongDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8066: Function EnglishShortMonthStrA( const Month : Integer) : AnsiString
8067: Function EnglishShortMonthStrU( const Month : Integer) : UnicodeString
8068: Function EnglishLongMonthStrA( const Month : Integer) : AnsiString
8069: Function EnglishLongMonthStrU( const Month : Integer) : UnicodeString
8070: Function EnglishShortDayOfWeekA( const S : AnsiString) : Integer
8071: Function EnglishShortDayOfWeekU( const S : UnicodeString) : Integer
8072: Function EnglishLongDayOfWeekA( const S : AnsiString) : Integer
8073: Function EnglishLongDayOfWeekU( const S : UnicodeString) : Integer
8074: Function EnglishShortMonthA( const S : AnsiString) : Integer
8075: Function EnglishShortMonthU( const S : UnicodeString) : Integer
8076: Function EnglishLongMonthA( const S : AnsiString) : Integer
8077: Function EnglishLongMonthU( const S : UnicodeString) : Integer
8078: Function RFC850DayOfWeekA( const S : AnsiString) : Integer
8079: Function RFC850DayOfWeekU( const S : UnicodeString) : Integer
8080: Function RFC1123DayOfWeekA( const S : AnsiString) : Integer
8081: Function RFC1123DayOfWeekU( const S : UnicodeString) : Integer
8082: Function RFCMonthA( const S : AnsiString) : Word
8083: Function RFCMonthU( const S : UnicodeString) : Word
8084: Function GMTTimeToRFC1123TimeA( const D : TDateTime; const IncludeSeconds:Boolean) : AnsiString
8085: Function GMTTimeToRFC1123TimeU( const D : TDateTime; const IncludeSeconds:Boolean) : UnicodeString
8086: Function GMTDateTimeToRFC1123DateTimeA(const D: TDateTime; const IncludeDayOfWeek:Bool):AnsiString;
8087: Function GMTDateTimeToRFC1123DateTimeU(const D:TDateTime;const IncludeDayOfWeek:Bool):UnicodeString;
8088: Function DateTimeToRFCDateTimeA( const D : TDateTime) : AnsiString
8089: Function DateTimeToRFCDateTimeU( const D : TDateTime) : UnicodeString
8090: Function NowAsRFCDateTimeA : AnsiString
8091: Function NowAsRFCDateTimeU : UnicodeString
8092: Function RFCDateTimeToGMTDateTime( const S : AnsiString) : TDateTime
8093: Function RFCDateTimeToDateTime( const S : AnsiString) : TDateTime
8094: Function RFCTimeZoneToGMTBias( const Zone : AnsiString) : Integer
8095: Function TimePeriodStr( const D : TDateTime) : AnsiString
8096: Procedure SelfTest
8097: end;
8098: //****************************************CFileUtils
8099: Function PathHasDriveLetterA( const Path : AnsiString) : Boolean
8100: Function PathHasDriveLetter( const Path : String) : Boolean
8101: Function PathIsDriveLetterA( const Path : AnsiString) : Boolean
8102: Function PathIsDriveLetter( const Path : String) : Boolean
8103: Function PathIsDriveRootA( const Path : AnsiString) : Boolean
8104: Function PathIsDriveRoot( const Path : String) : Boolean
8105: Function PathIsRootA( const Path : AnsiString) : Boolean
8106: Function PathIsRoot( const Path : String) : Boolean
8107: Function PathIsUNCPathA( const Path : AnsiString) : Boolean
8108: Function PathIsUNCPath( const Path : String) : Boolean
8109: Function PathIsAbsoluteA( const Path : AnsiString) : Boolean
8110: Function PathIsAbsolute( const Path : String) : Boolean
8111: Function PathIsDirectoryA( const Path : AnsiString) : Boolean
8112: Function PathIsDirectory( const Path : String) : Boolean
8113: Function PathInclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString
8114: Function PathInclSuffix( const Path : String; const PathSep : Char) : String
```

```
8115: Function PathExclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString
8116: Function PathExclSuffix( const Path : String; const PathSep : Char) : String
8117: Procedure PathEnsureSuffixA( var Path : AnsiString; const PathSep : Char)
8118: Procedure PathEnsureSuffix( var Path : String; const PathSep : Char)
8119: Procedure PathEnsureNoSuffixA( var Path : AnsiString; const PathSep : Char)
8120: Procedure PathEnsureNoSuffix( var Path : String; const PathSep : Char)
8121: //Function PathCanonicalA( const Path : AnsiString; const PathSep : Char) : AnsiString
8122: Function PathCanonical( const Path : String; const PathSep : Char) : String
8123: Function PathExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8124: Function PathExpand( const Path : String; const BasePath : String; const PathSep : Char) : String
8125: Function PathLeftElementA( const Path : AnsiString; const PathSep : Char) : AnsiString
8126: Function PathLeftElement( const Path : String; const PathSep : Char) : String
8127: Procedure PathSplitLeftElementA(const Path:AString;var LeftElement,RightPath:AString;const PathSep:Char);
8128: Procedure PathSplitLeftElement(const Path:String; var LeftElement,RightPath: String;const PathSep:Char);
8129: Procedure DecodeFilePathA(const FilePath:AnsiString; var Path,FileName:AnsiString;const PathSep:Char;
8130: Procedure DecodeFilePath( const FilePath : String; var Path, FileName : String; const PathSep : Char)
8131: Function FileNameValidA( const FileName : AnsiString) : AnsiString
8132: Function FileNameValid( const FileName : String) : String
8133: Function FilePathA(const FileName,Path:AnsiString;const BasePath:AnsiStr;const PathSep:Char):AnsiString;
8134: Function FilePath(const FileName, Path: String;const BasePath: String;const PathSep : Char) : String
8135: Function DirectoryExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8136: Function DirectoryExpand(const Path: String; const BasePath: String; const PathSep : Char) : String
8137: Function UnixPathToWinPath( const Path : AnsiString) : AnsiString
8138: Function WinPathToUnixPath( const Path : AnsiString) : AnsiString
8139: Procedure CCopyFile( const FileName, DestName : String)
8140: Procedure CMoveFile( const FileName, DestName : String)
8141: Function CDeleteFiles( const FileMask : String) : Boolean
8142: Function FileSeekEx(const FHandle:TFileHandle;const FileOffset:Int64; const FilePos:TFileSeekPos):Int64;
8143: Procedure FileCloseEx( const FileHandle : TFileHandle)
8144: Function FileExistsA( const FileName : AnsiString) : Boolean
8145:  Function CFileExists( const FileName : String) : Boolean
8146:  Function CFileGetSize( const FileName : String) : Int64
8147: Function FileGetDateTime( const FileName : String) : TDateTime
8148: Function FileGetDateTime2( const FileName : String) : TDateTime
8149: Function FileIsReadOnly( const FileName : String) : Boolean
8150: Procedure FileDeleteEx( const FileName : String)
8151: Procedure FileRenameEx( const OldFileName, NewFileName : String)
8152: Function ReadFileStrA( const FileName:AnsiString; const FileSharing : TFileSharing; const FileCreationMode
      : TFileCreationMode; const FileOpenWait : PFileOpenWait) : AnsiString
8153: Function DirectoryEntryExists( const Name : String) : Boolean
8154: Function DirectoryEntrySize( const Name : String) : Int64
8155:  Function CDirectoryExists( const DirectoryName : String) : Boolean
8156: Function DirectoryGetDateTime( const DirectoryName : String) : TDateTime
8157: Procedure CDirectoryCreate( const DirectoryName : String)
8158: Function GetFirstFileNameMatching( const FileMask : String) : String
8159: Function DirEntryGetAttr( const FileName : AnsiString) : Integer
8160: Function DirEntryIsDirectory( const FileName : AnsiString) : Boolean
8161: Function FileHasAttr( const FileName : String; const Attr : Word) : Boolean
8162:   AddTypeS('TLogicalDriveType', '( DriveRemovable, DriveFixed, DriveRemote, '
8163:    +'DriveCDRom, DriveRamDisk, DriveTypeUnknown )
8164: Function DriveIsValid( const Drive : Char) : Boolean
8165: Function DriveGetType( const Path : AnsiString) : TLogicalDriveType
8166: Function DriveFreeSpace( const Path : AnsiString) : Int64
8167:
8168: procedure SIRegister_cTimers(CL: TPSPascalCompiler);
8169: begin
8170:  AddClassN(FindClass('TOBJECT'),'ETimers
8171:  Const('TickFrequency','LongInt').SetInt( 1000);Function GetTick : LongWord
8172: Function TickDelta( const D1, D2 : LongWord) : Integer
8173: Function TickDeltaW( const D1, D2 : LongWord) : LongWord
8174:   AddTypeS('THPTimer', 'Int64
8175: Procedure StartTimer( var Timer : THPTimer)
8176: Procedure StopTimer( var Timer : THPTimer)
8177: Procedure ResumeTimer( var StoppedTimer : THPTimer)
8178: Procedure InitStoppedTimer( var Timer : THPTimer)
8179: Procedure InitElapsedTimer( var Timer : THPTimer; const Milliseconds : Integer)
8180: Function MillisecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Integer
8181: Function MicrosecondsElapsed( const Timer: THPTimer; const TimerRunning : Boolean) : Int64
8182: Procedure WaitMicroseconds( const MicroSeconds : Integer)
8183: Function GetHighPrecisionFrequency : Int64
8184: Function GetHighPrecisionTimerOverhead : Int64
8185: Procedure AdjustTimerForOverhead( var StoppedTimer : THPTimer; const Overhead : Int64)
8186: Procedure SelfTestCTimer
8187: end;
8188:
8189: procedure SIRegister_cRandom(CL: TPSPascalCompiler);
8190: begin
8191:  Function RandomSeed : LongWord
8192:  Procedure AddEntropy( const Value : LongWord)
8193:  Function RandomUniform : LongWord;
8194:  Function RandomUniform1( const N : Integer) : Integer;
8195:  Function RandomBoolean : Boolean
8196:  Function RandomByte : Byte
8197:  Function RandomByteNonZero : Byte
8198:  Function RandomWord : Word
8199:  Function RandomInt64 : Int64;
8200:  Function RandomInt641( const N : Int64) : Int64;
8201:  Function RandomHex( const Digits : Integer) : String
8202:  Function RandomFloat : Extended
```

```
8203:  Function RandomAlphaStr( const Length : Integer) : AnsiString
8204:  Function RandomPseudoWord( const Length : Integer) : AnsiString
8205:  Function RandomPassword(const MinL,MaxLength:Int;const CaseSens,UseSymbols,UseNumbers:Bool):AnsiString;
8206:  Function mwcRandomLongWord : LongWord
8207:  Function urnRandomLongWord : LongWord
8208:  Function moaRandomFloat : Extended
8209:  Function mwcRandomFloat : Extended
8210:  Function RandomNormalF : Extended
8211:  Procedure SelfTestCRandom
8212: end;
8213:
8214: procedure SIRegister_SynEditMiscProcs(CL: TPSPascalCompiler);
8215: begin
8216:  // PIntArray', '^TIntArray // will not work
8217:  Addtypes('TConvertTabsProc','function(const Line:AnsiString; TabWidth: integer):AnsiString
8218:  Addtypes('TConvertTabsProcEx','function(const Line:AnsiString; TabWidth: integer;var HasTabs: boolean):
      AnsiString
8219:  Function synMax( x, y : integer) : integer
8220:  Function synMin( x, y : integer) : integer
8221:  Function synMinMax( x, mi, ma : integer) : integer
8222:  Procedure synSwapInt( var l, r : integer)
8223:  Function synMaxPoint( const P1, P2 : TPoint) : TPoint
8224:  Function synMinPoint( const P1, P2 : TPoint) : TPoint
8225:  //Function synGetIntArray( Count : Cardinal; InitialValue : integer) : PIntArray
8226:  Procedure synInternalFillRect( dc : HDC; const rcPaint : TRect)
8227:  Function synGetBestConvertTabsProc( TabWidth : integer) : TConvertTabsProc
8228:  Function synConvertTabs( const Line : AnsiString; TabWidth : integer) : AnsiString
8229:  Function synGetBestConvertTabsProcEx( TabWidth : integer) : TConvertTabsProcEx
8230:  Function synConvertTabsEx(const Line:AnsiString;TabWidth:integer; var HasTabs:boolean):AnsiString;
8231:  Function synGetExpandedLength( const aStr : string; aTabWidth : integer) : integer
8232:  Function synCharIndex2CaretPos( Index, TabWidth : integer; const Line : string) : integer
8233:  Function synCaretPos2CharIndex(Position,TabWidth:int;const Line:string;var InsideTabChar:boolean):int;
8234:  Function synStrScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8235:  Function synStrRScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8236:   TStringType', '( stNone, stHalfNumAlpha, stHalfSymbol, stHalfKat'
8237:    +'akana,stWideNumAlpha,stWideSymbol,stWideKatakana,stHiragana,stIdeograph,stControl,stKashida )
8238:  ('C3_NONSPACING','LongInt').SetInt( 1);
8239:  'C3_DIACRITIC','LongInt').SetInt( 2);
8240:  'C3_VOWELMARK','LongInt').SetInt( 4);
8241:  ('C3_SYMBOL','LongInt').SetInt( 8);
8242:  ('C3_KATAKANA','LongWord').SetUInt( $0010);
8243:  ('C3_HIRAGANA','LongWord').SetUInt( $0020);
8244:  ('C3_HALFWIDTH','LongWord').SetUInt( $0040);
8245:  ('C3_FULLWIDTH','LongWord').SetUInt( $0080);
8246:  ('C3_IDEOGRAPH','LongWord').SetUInt( $0100);
8247:  ('C3_KASHIDA','LongWord').SetUInt( $0200);
8248:  ('C3_LEXICAL','LongWord').SetUInt( $0400);
8249:  ('C3_ALPHA','LongWord').SetUInt( $8000);
8250:  ('C3_NOTAPPLICABLE','LongInt').SetInt( 0);
8251:  Function synStrScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8252:  Function synStrRScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8253:  Function synIsStringType( Value : Word) : TStringType
8254:  Function synGetEOL( Line : PChar) : PChar
8255:  Function synEncodeString( s : string) : string
8256:  Function synDecodeString( s : string) : string
8257:  Procedure synFreeAndNil( var Obj: TObject)
8258:  Procedure synAssert( Expr : Boolean)
8259:  Function synLastDelimiter( const Delimiters, S : string) : Integer
8260:   TReplaceFlag', '( rfReplaceAll, rfIgnoreCase )
8261:   TReplaceFlags', 'set of TReplaceFlag )
8262:  Function synStringReplace(const S, OldPattern, NewPattern : string; Flags: TReplaceFlags) : string
8263:  Function synGetRValue( RGBValue : TColor) : byte
8264:  Function synGetGValue( RGBValue : TColor) : byte
8265:  Function synGetBValue( RGBValue : TColor) : byte
8266:  Function synRGB( r, g, b : Byte) : Cardinal
8267: //  THighlighterAttriProc', 'Function ( Highlighter : TSynCustomHigh'
8268:   // +'lighter; Attri:TSynHighlighterAttributes;UniqueAttriName:string;Params array of Pointer):Boolean;
8269:  //Function synEnumHighlighterAttris( Highlighter : TSynCustomHighlighter; SkipDuplicates : Boolean;
      HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer) : Boolean
8270:  Function synCalcFCS( const ABuf, ABufSize : Cardinal) : Word
8271:  Procedure synSynDrawGradient(const ACanvas:TCanvas;const AStartColor,
      AEndColor:TColor;ASteps:integer;const ARect : TRect; const AHorizontal : boolean)
8272: end;
8273:
8274:  Function GET_APPCOMMAND_LPARAM( lParam : LPARAM) : WORD
8275:  Function GET_DEVICE_LPARAM( lParam : LPARAM) : WORD
8276:  Function GET_KEYSTATE_LPARAM( lParam : LPARAM) : WORD
8277:
8278: procedure SIRegister_synautil(CL: TPSPascalCompiler);
8279: begin
8280:  Function STimeZoneBias : integer
8281:  Function TimeZone : string
8282:  Function Rfc822DateTime( t : TDateTime) : string
8283:  Function CDateTime( t : TDateTime) : string
8284:  Function SimpleDateTime( t : TDateTime) : string
8285:  Function AnsiCDateTime( t : TDateTime) : string
8286:  Function GetMonthNumber( Value : String) : integer
8287:  Function GetTimeFromStr( Value : string) : TDateTime
8288:  Function GetDateMDYFromStr( Value : string) : TDateTime
```

```
8289:  Function DecodeRfcDateTime( Value : string) : TDateTime
8290:  Function GetUTTime : TDateTime
8291:  Function SetUTTime( Newdt : TDateTime) : Boolean
8292:  Function SGetTick : LongWord
8293:  Function STickDelta( TickOld, TickNew : LongWord) : LongWord
8294:  Function CodeInt( Value : Word) : Ansistring
8295:  Function DecodeInt( const Value : Ansistring; Index : Integer) : Word
8296:  Function CodeLongInt( Value : LongInt) : Ansistring
8297:  Function DecodeLongInt( const Value : Ansistring; Index : Integer) : LongInt
8298:  Function DumpStr( const Buffer : Ansistring) : string
8299:  Function DumpExStr( const Buffer : Ansistring) : string
8300:  Procedure Dump( const Buffer : AnsiString; DumpFile : string)
8301:  Procedure DumpEx( const Buffer : AnsiString; DumpFile : string)
8302:  Function TrimSPLeft( const S : string) : string
8303:  Function TrimSPRight( const S : string) : string
8304:  Function TrimSP( const S : string) : string
8305:  Function SeparateLeft( const Value, Delimiter : string) : string
8306:  Function SeparateRight( const Value, Delimiter : string) : string
8307:  Function SGetParameter( const Value, Parameter : string) : string
8308:  Procedure ParseParametersEx( Value, Delimiter : string; const Parameters : TStrings)
8309:  Procedure ParseParameters( Value : string; const Parameters : TStrings)
8310:  Function IndexByBegin( Value : string; const List : TStrings) : integer
8311:  Function GetEmailAddr( const Value : string) : string
8312:  Function GetEmailDesc( Value : string) : string
8313:  Function CStrToHex( const Value : Ansistring) : string
8314:  Function CIntToBin( Value : Integer; Digits : Byte) : string
8315:  Function CBinToInt( const Value : string) : Integer
8316:  Function ParseURL( URL : string; var Prot, User, Pass, Host, Port, Path, Para:string):string
8317:  Function CReplaceString( Value, Search, Replace : AnsiString) : AnsiString
8318:  Function CRPosEx( const Sub, Value : string; From : integer) : Integer
8319:  Function CRPos( const Sub, Value : String) : Integer
8320:  Function FetchBin( var Value : string; const Delimiter : string) : string
8321:  Function CFetch( var Value : string; const Delimiter : string) : string
8322:  Function FetchEx( var Value : string; const Delimiter, Quotation : string) : string
8323:  Function IsBinaryString( const Value : AnsiString) : Boolean
8324:  Function PosCRLF( const Value : AnsiString; var Terminator : AnsiString) : integer
8325:  Procedure StringsTrim( const value : TStrings)
8326:  Function PosFrom( const SubStr, Value : String; From : integer) : integer
8327:  Function IncPoint( const p : ___pointer; Value : integer) : ___pointer
8328:  Function GetBetween( const PairBegin, PairEnd, Value : string) : string
8329:  Function CCountOfChar( const Value : string; aChr : char) : integer
8330:  Function UnquoteStr( const Value : string; Quote : Char) : string
8331:  Function QuoteStr( const Value : string; Quote : Char) : string
8332:  Procedure HeadersToList( const Value : TStrings)
8333:  Procedure ListToHeaders( const Value : TStrings)
8334:  Function SwapBytes( Value : integer) : integer
8335:  Function ReadStrFromStream( const Stream : TStream; len : integer) : AnsiString
8336:  Procedure WriteStrToStream( const Stream : TStream; Value : AnsiString)
8337:  Function GetTempFile( const Dir, prefix : AnsiString) : AnsiString
8338:  Function CPadString( const Value : AnsiString; len : integer; Pad : AnsiChar): AnsiString
8339:  Function CXorString( Indata1, Indata2 : AnsiString) : AnsiString
8340:  Function NormalizeHeader( Value : TStrings; var Index : Integer) : string
8341: end;
8342:
8343: procedure SIRegister_StCRC(CL: TPSPascalCompiler);
8344: begin
8345:  ('CrcBufSize','LongInt').SetInt( 2048);
8346:  Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8347:  Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8348:  Function Adler32OfFile( FileName : AnsiString) : LongInt
8349:  Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8350:  Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8351:  Function Crc16OfFile( FileName : AnsiString) : Cardinal
8352:  Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8353:  Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8354:  Function Crc32OfFile( FileName : AnsiString) : LongInt
8355:  Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8356:  Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8357:  Function InternetSumOfFile( FileName : AnsiString) : Cardinal
8358:  Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8359:  Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8360:  Function Kermit16OfFile( FileName : AnsiString) : Cardinal
8361: end;
8362:
8363: procedure SIRegister_ComObj(cl: TPSPascalCompiler);
8364: begin
8365:  function CreateOleObject(const ClassName: String): IDispatch;
8366:  function GetActiveOleObject(const ClassName: String): IDispatch;
8367:  function ProgIDToClassID(const ProgID: string): TGUID;
8368:  function ClassIDToProgID(const ClassID: TGUID): string;
8369:  function CreateClassID: string;
8370:  function CreateGUIDString: string;
8371:  function CreateGUIDID: string;
8372:  procedure OleError(ErrorCode: longint)
8373:  procedure OleCheck(Result: HResult);
8374: end;
8375:
8376:  Function xCreateOleObject( const ClassName : string) : Variant //or IDispatch
8377:  Function xGetActiveOleObject( const ClassName : string) : Variant
```

```
8378:  //Function DllGetClassObject( const CLSID : TCLSID; const IID : TIID; var Obj) : HResult
8379:  Function DllCanUnloadNow : HResult
8380:  Function DllRegisterServer : HResult
8381:  Function DllUnregisterServer : HResult
8382:  Function VarFromInterface( Unknown : IUnknown) : Variant
8383:  Function VarToInterface( const V : Variant) : IDispatch
8384:  Function VarToAutoObject( const V : Variant) : TAutoObject
8385:  //Procedure
      DispInvoke(Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer;Res:PVariant);
8386:  //Procedure DispInvokeError( Status : HResult; const ExcepInfo : TExcepInfo)
8387:  Procedure OleError( ErrorCode : HResult)
8388:  Procedure OleCheck( Result : HResult)
8389:  Function StringToClassID( const S : string) : TCLSID
8390:  Function ClassIDToString( const ClassID : TCLSID) : string
8391:  Function xProgIDToClassID( const ProgID : string) : TCLSID
8392:  Function xClassIDToProgID( const ClassID : TCLSID) : string
8393:  Function xWideCompareStr( const S1, S2 : WideString) : Integer
8394:  Function xWideSameStr( const S1, S2 : WideString) : Boolean
8395:  Function xGUIDToString( const ClassID : TGUID) : string
8396:  Function xStringToGUID( const S : string) : TGUID
8397:  Function xGetModuleName( Module : HMODULE) : string
8398:  Function xAcquireExceptionObject : TObject
8399:  Function xIfThen( AValue : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer
8400:  Function xUtf8Encode( const WS : WideString) : UTF8String
8401:  Function xUtf8Decode( const S : UTF8String) : WideString
8402:  Function xExcludeTrailingPathDelimiter( const S : string) : string
8403:  Function xIncludeTrailingPathDelimiter( const S : string) : string
8404:  Function XRTLHandleCOMException : HResult
8405:  Procedure XRTLCheckArgument( Flag : Boolean)
8406:  //Procedure XRTLCheckOutArgument( out Arg)
8407:  Procedure XRTLInterfaceConnect(const Source:IUnknown; const IID:TIID;const Sink:IUnknown;var
      Connection:Longint);
8408:  Procedure XRTLInterfaceDisconnect(const Source: IUnknown; const IID:TIID;var Connection : Longint)
8409:  Function XRTLRegisterActiveObject(const Unk:IUnknown;ClassID:TCLSID;Flags:DWORD;var
      RegisterCookie:Int):HResult
8410:  Function XRTLUnRegisterActiveObject( var RegisterCookie : Integer) : HResult
8411:  //Function XRTLGetActiveObject( ClassID : TCLSID; RIID : TIID; out Obj) : HResult
8412:  Procedure XRTLEnumActiveObjects( Strings : TStrings)
8413: function   XRTLDefaultCategoryManager: IUnknown;
8414: function   XRTLIsCategoryEmpty(CatID: TGUID; const CategoryManager: IUnknown = nil): Boolean;
8415: // ICatRegister helper functions
8416: function   XRTLCreateComponentCategory(CatID: TGUID; CatDescription: WideString;
8417:                                        LocaleID: TLCID = LOCALE_USER_DEFAULT;
8418:                                        const CategoryManager: IUnknown = nil): HResult;
8419: function   XRTLRemoveComponentCategory(CatID: TGUID; CatDescription: WideString;
8420:                                        LocaleID: TLCID = LOCALE_USER_DEFAULT;
8421:                                        const CategoryManager: IUnknown = nil): HResult;
8422: function   XRTLRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8423:                                        const CategoryManager: IUnknown = nil): HResult;
8424: function   XRTLUnRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8425:                                          const CategoryManager: IUnknown = nil): HResult;
8426: // ICatInformation helper functions
8427: function   XRTLGetCategoryDescription(CatID: TGUID; var CatDescription: WideString;
8428:                                       LocaleID: TLCID = LOCALE_USER_DEFAULT;
8429:                                       const CategoryManager: IUnknown = nil): HResult;
8430: function   XRTLGetCategoryList(Strings: TStrings; LocaleID: TLCID = LOCALE_USER_DEFAULT;
8431:                                const CategoryManager: IUnknown = nil): HResult;
8432: function   XRTLGetCategoryCLSIDList(CatID: TGUID; Strings: TStrings;
8433:                                     const CategoryManager: IUnknown = nil): HResult;
8434: function   XRTLGetCategoryProgIDList(CatID: TGUID; Strings: TStrings;
8435:                                       const CategoryManager: IUnknown = nil): HResult;
8436: function XRTLFetch(var AInput: WideString; const ADelim: WideString = ' ';
8437:                    const ADelete: Boolean = True): WideString;
8438: function XRTLRPos(const ASub, AIn: WideString; AStart: Integer = -1): Integer;
8439: Function XRTLGetVariantAsString( const Value : Variant) : string
8440: Function XRTLDateTimeToTimeZoneTime( DT : TDateTime; TimeZone : TXRTLTimeZone) : TDateTime
8441: Function XRTLGetTimeZones : TXRTLTimeZones
8442: Function XFileTimeToDateTime( FileTime : TFileTime) : TDateTime
8443: Function DateTimeToFileTime( DateTime : TDateTime) : TFileTime
8444: Function GMTNow : TDateTime
8445: Function GMTToLocalTime( GMT : TDateTime) : TDateTime
8446: Function LocalTimeToGMT( LocalTime : TDateTime) : TDateTime
8447: Procedure XRTLNotImplemented
8448: Procedure XRTLRaiseError( E : Exception)
8449: Procedure XRTLInvalidOperation( ClassName:string; OperationName:string; Description: string)
8450:
8451:
8452: procedure SIRegister_xrtl_util_Value(CL: TPSPascalCompiler);
8453: begin
8454:   SIRegister_IXRTLValue(CL);
8455:   SIRegister_TXRTLValue(CL);
8456:   //AddTypeS('PXRTLValueArray', '^TXRTLValueArray // will not work
8457:   AddTypeS('TXRTLValueArray', 'array of IXRTLValue
8458: Function XRTLValue( const AValue : Cardinal) : IXRTLValue;
8459: Function XRTLSetValue( const IValue : IXRTLValue; const AValue : Cardinal) : Cardinal;
8460: Function XRTLGetAsCardinal( const IValue : IXRTLValue) : Cardinal
8461: Function XRTLGetAsCardinalDef( const IValue : IXRTLValue; const DefValue : Cardinal) : Cardinal
8462: Function XRTLValue1( const AValue : Integer) : IXRTLValue;
8463: Function XRTLSetValue1( const IValue : IXRTLValue; const AValue : Integer) : Integer;
```

```
8464: Function XRTLGetAsInteger( const IValue : IXRTLValue) : Integer
8465: Function XRTLGetAsIntegerDef( const IValue : IXRTLValue; const DefValue : Integer) : Integer
8466: Function XRTLValue2( const AValue : Int64) : IXRTLValue;
8467: Function XRTLSetValue2( const IValue : IXRTLValue; const AValue : Int64) : Int64;
8468: Function XRTLGetAsInt64( const IValue : IXRTLValue) : Int64
8469: Function XRTLGetAsInt64Def( const IValue : IXRTLValue; const DefValue : Int64) : Int64
8470: Function XRTLValue3( const AValue : Single) : IXRTLValue;
8471: Function XRTLSetValue3( const IValue : IXRTLValue; const AValue : Single) : Single;
8472: Function XRTLGetAsSingle( const IValue : IXRTLValue) : Single
8473: Function XRTLGetAsSingleDef( const IValue : IXRTLValue; const DefValue : Single) : Single
8474: Function XRTLValue4( const AValue : Double) : IXRTLValue;
8475: Function XRTLSetValue4( const IValue : IXRTLValue; const AValue : Double) : Double;
8476: Function XRTLGetAsDouble( const IValue : IXRTLValue) : Double
8477: Function XRTLGetAsDoubleDef( const IValue : IXRTLValue; const DefValue : Double) : Double
8478: Function XRTLValue5( const AValue : Extended) : IXRTLValue;
8479: Function XRTLSetValue5( const IValue : IXRTLValue; const AValue : Extended) : Extended;
8480: Function XRTLGetAsExtended( const IValue : IXRTLValue) : Extended
8481: Function XRTLGetAsExtendedDef( const IValue : IXRTLValue; const DefValue : Extended) : Extended
8482: Function XRTLValue6( const AValue : IInterface) : IXRTLValue;
8483: Function XRTLSetValue6( const IValue : IXRTLValue; const AValue : IInterface) : IInterface;
8484: Function XRTLGetAsInterface( const IValue : IXRTLValue) : IInterface;
8485:  //Function XRTLGetAsInterface1( const IValue : IXRTLValue; out Obj) : IInterface;
8486: Function XRTLGetAsInterfaceDef( const IValue : IXRTLValue; const DefValue : IInterface) : IInterface;
8487: Function XRTLValue7( const AValue : WideString) : IXRTLValue;
8488: Function XRTLSetValue7( const IValue : IXRTLValue; const AValue : WideString) : WideString;
8489: Function XRTLGetAsWideString( const IValue : IXRTLValue) : WideString
8490: Function XRTLGetAsWideStringDef( const IValue : IXRTLValue; const DefValue : WideString) : WideString
8491: Function XRTLValue8( const AValue : TObject; const AOwnValue : Boolean) : IXRTLValue;
8492: Function XRTLSetValue8( const IValue : IXRTLValue; const AValue : TObject) : TObject;
8493: Function XRTLGetAsObject( const IValue : IXRTLValue; const ADetachOwnership : Boolean) : TObject;
8494: Function XRTLGetAsObjectDef(const IValue:IXRTLValue;const DefValue:TObject;const
      ADetachOwnership:Boolean):TObject;
8495: //Function XRTLValue9( const AValue : __Pointer) : IXRTLValue;
8496: //Function XRTLSetValue9( const IValue : IXRTLValue; const AValue : __Pointer) : __Pointer;
8497: //Function XRTLGetAsPointer( const IValue : IXRTLValue) : __Pointer
8498: //Function XRTLGetAsPointerDef( const IValue : IXRTLValue; const DefValue : __Pointer) : __Pointer
8499: Function XRTLValueV( const AValue : Variant) : IXRTLValue;
8500: Function XRTLSetValueV( const IValue : IXRTLValue; const AValue : Variant) : Variant;
8501: Function XRTLGetAsVariant( const IValue : IXRTLValue) : Variant
8502: Function XRTLGetAsVariantDef( const IValue : IXRTLValue; const DefValue : Variant) : Variant
8503: Function XRTLValue10( const AValue : Currency) : IXRTLValue;
8504: Function XRTLSetValue10( const IValue : IXRTLValue; const AValue : Currency) : Currency;
8505: Function XRTLGetAsCurrency( const IValue : IXRTLValue) : Currency
8506: Function XRTLGetAsCurrencyDef( const IValue : IXRTLValue; const DefValue : Currency) : Currency
8507: Function XRTLValue11( const AValue : Comp) : IXRTLValue;
8508: Function XRTLSetValue11( const IValue : IXRTLValue; const AValue : Comp) : Comp;
8509: Function XRTLGetAsComp( const IValue : IXRTLValue) : Comp
8510: Function XRTLGetAsCompDef( const IValue : IXRTLValue; const DefValue : Comp) : Comp
8511: Function XRTLValue12( const AValue : TClass) : IXRTLValue;
8512: Function XRTLSetValue12( const IValue : IXRTLValue; const AValue : TClass) : TClass;
8513: Function XRTLGetAsClass( const IValue : IXRTLValue) : TClass
8514: Function XRTLGetAsClassDef( const IValue : IXRTLValue; const DefValue : TClass) : TClass
8515: Function XRTLValue13( const AValue : TGUID) : IXRTLValue;
8516: Function XRTLSetValue13( const IValue : IXRTLValue; const AValue : TGUID) : TGUID;
8517: Function XRTLGetAsGUID( const IValue : IXRTLValue) : TGUID
8518: Function XRTLGetAsGUIDDef( const IValue : IXRTLValue; const DefValue : TGUID) : TGUID
8519: Function XRTLValue14( const AValue : Boolean) : IXRTLValue;
8520: Function XRTLSetValue14( const IValue : IXRTLValue; const AValue : Boolean) : Boolean;
8521: Function XRTLGetAsBoolean( const IValue : IXRTLValue) : Boolean
8522: Function XRTLGetAsBooleanDef( const IValue : IXRTLValue; const DefValue : Boolean) : Boolean
8523: end;
8524:
8525: //****************************unit uPSI_GR32;****************************************
8526:
8527:  Function Color32( WinColor : TColor) : TColor32;
8528:  Function Color321( R, G, B : Byte; A : Byte) : TColor32;
8529:  Function Color322( Index : Byte; var Palette : TPalette32) : TColor32;
8530:  Function Gray32( Intensity : Byte; Alpha : Byte) : TColor32
8531:  Function WinColor( Color32 : TColor32) : TColor
8532:  Function ArrayOfColor32( Colors : array of TColor32) : TArrayOfColor32
8533:  Procedure Color32ToRGB( Color32 : TColor32; var R, G, B : Byte)
8534:  Procedure Color32ToRGBA( Color32 : TColor32; var R, G, B, A : Byte)
8535:  Function Color32Components( R, G, B, A : Boolean) : TColor32Components
8536:  Function RedComponent( Color32 : TColor32) : Integer
8537:  Function GreenComponent( Color32 : TColor32) : Integer
8538:  Function BlueComponent( Color32 : TColor32) : Integer
8539:  Function AlphaComponent( Color32 : TColor32) : Integer
8540:  Function Intensity( Color32 : TColor32) : Integer
8541:  Function SetAlpha( Color32 : TColor32; NewAlpha : Integer) : TColor32
8542:  Function HSLtoRGB( H, S, L : Single) : TColor32;
8543:  Procedure RGBtoHSL( RGB : TColor32; out H, S, L : Single);
8544:  Function HSLtoRGB1( H, S, L : Integer) : TColor32;
8545:  Procedure RGBtoHSL1( RGB : TColor32; out H, S, L : Byte);
8546:  Function WinPalette( const P : TPalette32) : HPALETTE
8547:  Function FloatPoint( X, Y : Single) : TFloatPoint;
8548:  Function FloatPoint1( const P : TPoint) : TFloatPoint;
8549:  Function FloatPoint2( const FXP : TFixedPoint) : TFloatPoint;
8550:  Function FixedPoint( X, Y : Integer) : TFixedPoint;
8551:  Function FixedPoint1( X, Y : Single) : TFixedPoint;
```

```
8552:  Function FixedPoint2( const P : TPoint) : TFixedPoint;
8553:  Function FixedPoint3( const FP : TFloatPoint) : TFixedPoint;
8554:   AddTypeS('TRectRounding', '( rrClosest, rrOutside, rrInside )
8555:  Function MakeRect( const L, T, R, B : Integer) : TRect;
8556:  Function MakeRect1( const FR : TFloatRect; Rounding : TRectRounding) : TRect;
8557:  Function MakeRect2( const FXR : TRect; Rounding : TRectRounding) : TRect;
8558:  Function GFixedRect( const L, T, R, B : TFixed) : TRect;
8559:  Function FixedRect1( const ARect : TRect) : TRect;
8560:  Function FixedRect2( const FR : TFloatRect) : TRect;
8561:  Function GFloatRect( const L, T, R, B : TFloat) : TFloatRect;
8562:  Function FloatRect1( const ARect : TRect) : TFloatRect;
8563:  Function FloatRect2( const FXR : TRect) : TFloatRect;
8564:  Function GIntersectRect( out Dst : TRect; const R1, R2 : TRect) : Boolean;
8565:  Function IntersectRect1( out Dst : TFloatRect; const FR1, FR2 : TFloatRect) : Boolean;
8566:  Function GUnionRect( out Rect : TRect; const R1, R2 : TRect) : Boolean;
8567:  Function UnionRect1( out Rect : TFloatRect; const R1, R2 : TFloatRect) : Boolean;
8568:  Function GEqualRect( const R1, R2 : TRect) : Boolean;
8569:  Function EqualRect1( const R1, R2 : TFloatRect) : Boolean;
8570:  Procedure GInflateRect( var R : TRect; Dx, Dy : Integer);
8571:  Procedure InflateRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8572:  Procedure GOffsetRect( var R : TRect; Dx, Dy : Integer);
8573:  Procedure OffsetRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8574:  Function IsRectEmpty( const R : TRect) : Boolean;
8575:  Function IsRectEmpty1( const FR : TFloatRect) : Boolean;
8576:  Function GPtInRect( const R : TRect; const P : TPoint) : Boolean;
8577:  Function PtInRect1( const R : TFloatRect; const P : TPoint) : Boolean;
8578:  Function PtInRect2( const R : TRect; const P : TFloatPoint) : Boolean;
8579:  Function PtInRect3( const R : TFloatRect; const P : TFloatPoint) : Boolean;
8580:  Function EqualRectSize( const R1, R2 : TRect) : Boolean;
8581:  Function EqualRectSize1( const R1, R2 : TFloatRect) : Boolean;
8582:  Function MessageBeep( uType : UINT) : BOOL
8583:  Function ShowCursor( bShow : BOOL) : Integer
8584:  Function SetCursorPos( X, Y : Integer) : BOOL
8585:  Function SetCursor( hCursor : HICON) : HCURSOR
8586:  Function GetCursorPos( var lpPoint : TPoint) : BOOL
8587:  //Function ClipCursor( lpRect : PRect) : BOOL
8588:  Function GetClipCursor( var lpRect : TRect) : BOOL
8589:  Function GetCursor : HCURSOR
8590:  Function CreateCaret( hWnd : HWND; hBitmap : HBITMAP; nWidth, nHeight : Integer) : BOOL
8591:  Function GetCaretBlinkTime : UINT
8592:  Function SetCaretBlinkTime( uMSeconds : UINT) : BOOL
8593:  Function DestroyCaret : BOOL
8594:  Function HideCaret( hWnd : HWND) : BOOL
8595:  Function ShowCaret( hWnd : HWND) : BOOL
8596:  Function SetCaretPos( X, Y : Integer) : BOOL
8597:  Function GetCaretPos( var lpPoint : TPoint) : BOOL
8598:  Function ClientToScreen( hWnd : HWND; var lpPoint : TPoint) : BOOL
8599:  Function ScreenToClient( hWnd : HWND; var lpPoint : TPoint) : BOOL
8600:  Function MapWindowPoints(hWndFrom,hWndTo:HWND; var lpPoints, cPoints : UINT) : Integer
8601:  Function WindowFromPoint( Point : TPoint) : HWND
8602:  Function ChildWindowFromPoint( hWndParent : HWND; Point : TPoint) : HWND
8603:
8604:
8605: procedure SIRegister_GR32_Math(CL: TPSPascalCompiler);
8606: begin
8607:  Function FixedFloor( A : TFixed) : Integer
8608:  Function FixedCeil( A : TFixed) : Integer
8609:  Function FixedMul( A, B : TFixed) : TFixed
8610:  Function FixedDiv( A, B : TFixed) : TFixed
8611:  Function OneOver( Value : TFixed) : TFixed
8612:  Function FixedRound( A : TFixed) : Integer
8613:  Function FixedSqr( Value : TFixed) : TFixed
8614:  Function FixedSqrtLP( Value : TFixed) : TFixed
8615:  Function FixedSqrtHP( Value : TFixed) : TFixed
8616:  Function FixedCombine( W, X, Y : TFixed) : TFixed
8617:  Procedure GRSinCos( const Theta : TFloat; out Sin, Cos : TFloat);
8618:  Procedure GRSinCos1( const Theta, Radius : Single; out Sin, Cos : Single);
8619:  Function GRHypot( const X, Y : TFloat) : TFloat;
8620:  Function Hypot1( const X, Y : Integer) : Integer;
8621:  Function FastSqrt( const Value : TFloat) : TFloat
8622:  Function FastSqrtBab1( const Value : TFloat) : TFloat
8623:  Function FastSqrtBab2( const Value : TFloat) : TFloat
8624:  Function FastInvSqrt( const Value : Single) : Single;
8625:  Function MulDiv( Multiplicand, Multiplier, Divisor : Integer) : Integer
8626:  Function GRIsPowerOf2( Value : Integer) : Boolean
8627:  Function PrevPowerOf2( Value : Integer) : Integer
8628:  Function NextPowerOf2( Value : Integer) : Integer
8629:  Function Average( A, B : Integer) : Integer
8630:  Function GRSign( Value : Integer) : Integer
8631:  Function FloatMod( x, y : Double) : Double
8632: end;
8633:
8634: procedure SIRegister_GR32_LowLevel(CL: TPSPascalCompiler);
8635: begin
8636:  Function Clamp( const Value : Integer) : Integer;
8637:  Procedure GRFillWord( var X, Count : Cardinal; Value : Longword)
8638:  Function StackAlloc( Size : Integer) : Pointer
8639:  Procedure StackFree( P : Pointer)
8640:  Procedure Swap( var A, B : Pointer);
```

```
8641:  Procedure Swap1( var A, B : Integer);
8642:  Procedure Swap2( var A, B : TFixed);
8643:  Procedure Swap3( var A, B : TColor32);
8644:  Procedure TestSwap( var A, B : Integer);
8645:  Procedure TestSwap1( var A, B : TFixed);
8646:  Function TestClip( var A, B : Integer; const Size : Integer) : Boolean;
8647:  Function TestClip1( var A, B : Integer; const Start, Stop : Integer) : Boolean;
8648:  Function GRConstrain( const Value, Lo, Hi : Integer) : Integer;
8649:  Function Constrain1( const Value, Lo, Hi : Single) : Single;
8650:  Function SwapConstrain( const Value:Integer; Constrain1,Constrain2:Integer) : Integer
8651:  Function GRMin( const A, B, C : Integer) : Integer;
8652:  Function GRMax( const A, B, C : Integer) : Integer;
8653:  Function Clamp( Value, Max : Integer) : Integer;
8654:  Function Clamp1( Value, Min, Max : Integer) : Integer;
8655:  Function Wrap( Value, Max : Integer) : Integer;
8656:  Function Wrap1( Value, Min, Max : Integer) : Integer;
8657:  Function Wrap3( Value, Max : Single) : Single;;
8658:  Function WrapPow2( Value, Max : Integer) : Integer;
8659:  Function WrapPow21( Value, Min, Max : Integer) : Integer;
8660:  Function Mirror( Value, Max : Integer) : Integer;
8661:  Function Mirror1( Value, Min, Max : Integer) : Integer;
8662:  Function MirrorPow2( Value, Max : Integer) : Integer;
8663:  Function MirrorPow21( Value, Min, Max : Integer) : Integer;
8664:  Function GetOptimalWrap( Max : Integer) : TWrapProc;
8665:  Function GetOptimalWrap1( Min, Max : Integer) : TWrapProcEx;
8666:  Function GetOptimalMirror( Max : Integer) : TWrapProc;
8667:  Function GetOptimalMirror1( Min, Max : Integer) : TWrapProcEx;
8668:  Function GetWrapProc( WrapMode : TWrapMode) : TWrapProc;
8669:  Function GetWrapProc1( WrapMode : TWrapMode; Max : Integer) : TWrapProc;
8670:  Function GetWrapProcEx( WrapMode : TWrapMode) : TWrapProcEx;
8671:  Function GetWrapProcEx1( WrapMode : TWrapMode; Min, Max : Integer):TWrapProcEx;
8672:  Function Div255( Value : Cardinal) : Cardinal
8673:  Function SAR_4( Value : Integer) : Integer
8674:  Function SAR_8( Value : Integer) : Integer
8675:  Function SAR_9( Value : Integer) : Integer
8676:  Function SAR_11( Value : Integer) : Integer
8677:  Function SAR_12( Value : Integer) : Integer
8678:  Function SAR_13( Value : Integer) : Integer
8679:  Function SAR_14( Value : Integer) : Integer
8680:  Function SAR_15( Value : Integer) : Integer
8681:  Function SAR_16( Value : Integer) : Integer
8682:  Function ColorSwap( WinColor : TColor) : TColor32
8683: end;
8684:
8685: procedure SIRegister_GR32_Filters(CL: TPSPascalCompiler);
8686: begin
8687:  AddTypeS('TLogicalOperator', '( loXOR, loAND, loOR )
8688:  Procedure CopyComponents( Dst, Src : TCustomBitmap32; Components : TColor32Components);
8689:  Procedure CopyComponents1(Dst:TCustomBmap32;DstX,
      DstY:Int;Src:TCustomBmap32;SrcRect:TRect;Components:TColor32Comp;
8690:  Procedure AlphaToGrayscale( Dst, Src : TCustomBitmap32)
8691:  Procedure ColorToGrayscale( Dst, Src : TCustomBitmap32; PreserveAlpha : Boolean)
8692:  Procedure IntensityToAlpha( Dst, Src : TCustomBitmap32)
8693:  Procedure Invert( Dst, Src : TCustomBitmap32; Components : TColor32Components)
8694:  Procedure InvertRGB( Dst, Src : TCustomBitmap32)
8695:  Procedure ApplyLUT( Dst, Src : TCustomBitmap32; const LUT : TLUT8; PreserveAlpha : Boolean)
8696:  Procedure ChromaKey( ABitmap : TCustomBitmap32; TrColor : TColor32)
8697:  Function CreateBitmask( Components : TColor32Components) : TColor32
8698:  Procedure ApplyBitmask(Dst: TCustomBitmap32; DstX,DstY:Integer; Src:TCustomBitmap32; SrcRect : TRect;
      Bitmask : TColor32; LogicalOperator : TLogicalOperator);
8699:  Procedure
      ApplyBitmask1(ABitmap:TCustomBitmap32;ARect:TRect;Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8700:  Procedure CheckParams( Dst, Src : TCustomBitmap32; ResizeDst : Boolean)
8701: end;
8702:
8703:
8704: procedure SIRegister_JclNTFS(CL: TPSPascalCompiler);
8705: begin
8706:   AddClassN(FindClass('TOBJECT'),'EJclNtfsError
8707:   AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )
8708:  Function NtfsGetCompression( const FileName : string; var State : Short) : Boolean;
8709:  Function NtfsGetCompression1( const FileName : string) : TFileCompressionState;
8710:  Function NtfsSetCompression( const FileName : string; const State : Short) : Boolean
8711:  Procedure NtfsSetFileCompression( const FileName : string; const State : TFileCompressionState)
8712:  Procedure NtfsSetDirectoryTreeCompression(const Directory: string; const State : TFileCompressionState)
8713:  Procedure NtfsSetDefaultFileCompression(const Directory: string; const State:TFileCompressionState)
8714:  Procedure NtfsSetPathCompression(const Path:string;const State:TFileCompressionState;Recursive:Boolean;
8715:   //AddTypeS('TNtfsAllocRanges', 'record Entries : Integer; Data : PFileAlloca'
8716:    //+'tedRangeBuffer; MoreData : Boolean; end
8717:  Function NtfsSetSparse( const FileName : string) : Boolean
8718:  Function NtfsZeroDataByHandle( const Handle : THandle; const First, Last : Int64) : Boolean
8719:  Function NtfsZeroDataByName( const FileName : string; const First, Last : Int64) : Boolean
8720:  //Function NtfsQueryAllocRanges(const FileName:string;Offset,Count:Int64;var
      Ranges:TNtfsAllocRanges):Boolean;
8721:  //Function NtfsGetAllocRangeEntry( const Ranges : TNtfsAllocRanges;
      Index:Integer):TFileAllocatedRangeBuffer
8722:  Function NtfsSparseStreamsSupported( const Volume : string) : Boolean
8723:  Function NtfsGetSparse( const FileName : string) : Boolean
8724:  Function NtfsDeleteReparsePoint( const FileName : string; ReparseTag : DWORD) : Boolean
```

```
8725:  Function NtfsSetReparsePoint( const FileName : string; var ReparseData, Size : Longword) : Boolean
8726:  //Function NtfsGetReparsePoint(const FileName:string; var ReparseData:TReparseGuidDataBuffer):Boolean
8727:  Function NtfsGetReparseTag( const Path : string; var Tag : DWORD) : Boolean
8728:  Function NtfsReparsePointsSupported( const Volume : string) : Boolean
8729:  Function NtfsFileHasReparsePoint( const Path : string) : Boolean
8730:  Function NtfsIsFolderMountPoint( const Path : string) : Boolean
8731:  Function NtfsMountDeviceAsDrive( const Device : string; Drive : Char) : Boolean
8732:  Function NtfsMountVolume( const Volume : Char; const MountPoint : string) : Boolean
8733:   AddTypeS('TOpLock', '( olExclusive, olReadOnly, olBatch, olFilter )
8734:  Function NtfsOpLockAckClosePending( Handle : THandle; Overlapped : TOverlapped) : Boolean
8735:  Function NtfsOpLockBreakAckNo2( Handle : THandle; Overlapped : TOverlapped) : Boolean
8736:  Function NtfsOpLockBreakAcknowledge( Handle : THandle; Overlapped : TOverlapped) : Boolean
8737:  Function NtfsOpLockBreakNotify( Handle : THandle; Overlapped : TOverlapped) : Boolean
8738:  Function NtfsRequestOpLock( Handle : THandle; Kind : TOpLock; Overlapped : TOverlapped) : Boolean
8739:  Function NtfsCreateJunctionPoint( const Source, Destination : string) : Boolean
8740:  Function NtfsDeleteJunctionPoint( const Source : string) : Boolean
8741:  Function NtfsGetJunctionPointDestination( const Source : string; var Destination : string) : Boolean
8742:   AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
8743:    +'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile )
8744:   AddTypeS('TStreamIds', 'set of TStreamId
8745:   AddTypeS('TInternalFindStreamData', 'record FileHandle : THandle; Context '
8746:    +': ___Pointer; StreamIds : TStreamIds; end
8747:   AddTypeS('TFindStreamData', 'record internal : TInternalFindStreamData; At'
8748:    +'tributes : DWORD; StreamID : TStreamId; Name : WideString; Size : Int64; end
8749:  Function NtfsFindFirstStream(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Boolean;
8750:  Function NtfsFindNextStream( var Data : TFindStreamData) : Boolean
8751:  Function NtfsFindStreamClose( var Data : TFindStreamData) : Boolean
8752:  Function NtfsCreateHardLink( const LinkFileName, ExistingFileName : string) : Boolean
8753:   AddTypeS('TNtfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end
8754:  Function NtfsGetHardLinkInfo( const FileName : string; var Info : TNtfsHardLinkInfo) : Boolean
8755:  Function NtfsFindHardLinks(const Path:string;const FileIndexHigh,FileIndexLow:Cardinal;const
      List:TStrings):Bool;
8756:  Function NtfsDeleteHardLinks( const FileName : string) : Boolean
8757:  Function JclAppInstances : TJclAppInstances;
8758:  Function JclAppInstances1( const UniqueAppIdGuidStr : string) : TJclAppInstances;
8759:  Function ReadMessageCheck( var Message: TMessage;const IgnoredOriginatorWnd: HWND) : TJclAppInstDataKind
8760:  Procedure ReadMessageData( const Message : TMessage; var Data : ___Pointer; var Size : Integer)
8761:  Procedure ReadMessageString( const Message : TMessage; var S : string)
8762:  Procedure ReadMessageStrings( const Message : TMessage; const Strings : TStrings)
8763:
8764:
8765: (*----------------------------------------------------------------------------*)
8766: procedure SIRegister_JclGraphics(CL: TPSPascalCompiler);
8767: begin
8768:   FindClass('TOBJECT'),'EJclGraphicsError
8769:   TDynDynIntegerArrayArray', 'array of TDynIntegerArray
8770:   TDynPointArray', 'array of TPoint
8771:   TDynDynPointArrayArray', 'array of TDynPointArray
8772:   TPointF', 'record X : Single; Y : Single; end
8773:   TDynPointArrayF', 'array of TPointF
8774:   TDrawMode2', '( dmOpaque, dmBlend )
8775:   TStretchFilter2', '( sfNearest, sfLinear, sfSpline )
8776:   TConversionKind', '( ckRed, ckGreen, ckBlue, ckAlpha, ckUniformRGB, ckWeightedRGB )
8777:   TResamplingFilter', '( rfBox, rfTriangle, rfHermite, rfBell, rfSpline, rfLanczos3, rfMitchell )
8778:   TMatrix3d', 'record array[0..2,0..2] of extended end
8779:   TDynDynPointArrayArrayF', 'array of TDynPointArrayF
8780:   TScanLine', 'array of Integer
8781:   TScanLines', 'array of TScanLine
8782:   TColorChannel', '( ccRed, ccGreen, ccBlue, ccAlpha )
8783:   TGradientDirection', '( gdVertical, gdHorizontal )
8784:   TPolyFillMode', '( fmAlternate, fmWinding )
8785:   TJclRegionCombineOperator', '( coAnd, coDiff, coOr, coXor )
8786:   TJclRegionBitmapMode', '( rmInclude, rmExclude )
8787:   TJclRegionKind', '( rkNull, rkSimple, rkComplex, rkError )
8788:   SIRegister_TJclDesktopCanvas(CL);
8789:   FindClass('TOBJECT'),'TJclRegion
8790:   SIRegister_TJclRegionInfo(CL);
8791:   SIRegister_TJclRegion(CL);
8792:   SIRegister_TJclThreadPersistent(CL);
8793:   SIRegister_TJclCustomMap(CL);
8794:   SIRegister_TJclBitmap32(CL);
8795:   SIRegister_TJclByteMap(CL);
8796:   SIRegister_TJclTransformation(CL);
8797:   SIRegister_TJclLinearTransformation(CL);
8798:  Procedure Stretch(NewWidth,
      NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Source:TGraphic;Target:TBitmap);
8799:  Procedure Stretch1(NewWidth,NewHeight:Cardinal;Filter:TResamplingFilter;Radius:Single;Bitmap:TBitmap);
8800:  Procedure DrawBitmap( DC : HDC; Bitmap : HBitMap; X, Y, Width, Height : Integer)
8801:  Function GetAntialiasedBitmap( const Bitmap : TBitmap) : TBitmap
8802:  Procedure BitmapToJPeg( const FileName : string)
8803:  Procedure JPegToBitmap( const FileName : string)
8804:  Function ExtractIconCount( const FileName : string) : Integer
8805:  Function BitmapToIconJ( Bitmap : HBITMAP; cx, cy : Integer) : HICON
8806:  Function IconToBitmapJ( Icon : HICON) : HBITMAP
8807:  Procedure BlockTransfer( Dst : TJclBitmap32; DstX : Integer; DstY : Integer; Src : TJclBitmap32; SrcRect
      : TRect; CombineOp : TDrawMode)
8808:  Procedure StretchTransfer(Dst:TJclBitmap32;
      DstRect:TRect;Src:TJclBitmap32;SrcRect:TRect;StretchFilter:TStretchFilter; CombineOp : TDrawMode)
8809:  Procedure Transform( Dst, Src : TJclBitmap32; SrcRect : TRect; Transformation : TJclTransformation)
```

```
8810:  Procedure SetBorderTransparent( ABitmap : TJclBitmap32; ARect : TRect)
8811:  Function FillGradient( DC : HDC; ARect : TRect; ColorCount : Integer; StartColor, EndColor : TColor;
      ADirection : TGradientDirection) : Boolean;
8812:  Function CreateRegionFromBitmap(Bitmap: TBitmap; RegionColor:TColor;
      RegionBitmapMode:TJclRegionBitmapMode): HRGN
8813:  Procedure ScreenShot( bm : TBitmap; Left, Top, Width, Height : Integer; Window : HWND);
8814:  Procedure ScreenShot1( bm : TBitmap);
8815:  Procedure PolyLineTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8816:  Procedure PolyLineAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8817:  Procedure PolyLineFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8818:  Procedure PolygonTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8819:  Procedure PolygonAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32)
8820:  Procedure PolygonFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32)
8821:  Procedure PolyPolygonTS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8822:  Procedure PolyPolygonAS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArray;Color:TColor32);
8823:  Procedure PolyPolygonFS(Bitmap:TJclBitmap32;const Points:TDynDynPointArrayArrayF;Color:TColor32);
8824:  Procedure AlphaToGrayscale( Dst, Src : TJclBitmap32)
8825:  Procedure IntensityToAlpha( Dst, Src : TJclBitmap32)
8826:  Procedure Invert( Dst, Src : TJclBitmap32)
8827:  Procedure InvertRGB( Dst, Src : TJclBitmap32)
8828:  Procedure ColorToGrayscale( Dst, Src : TJclBitmap32)
8829:  Procedure ApplyLUT( Dst, Src : TJclBitmap32; const LUT : TLUT8)
8830:  Procedure SetGamma( Gamma : Single)
8831: end;
8832:
8833: (*----------------------------------------------------------------------------*)
8834: procedure SIRegister_JclSynch(CL: TPSPascalCompiler);
8835: begin
8836:  Function LockedAdd( var Target : Integer; Value : Integer) : Integer
8837:  Function LockedCompareExchange( var Target : Integer; Exch, Comp : Integer) : Integer;
8838:  Function LockedCompareExchange1( var Target : ___Pointer; Exch, Comp : ___Pointer) : Pointer;
8839:  Function LockedDec( var Target : Integer) : Integer
8840:  Function LockedExchange( var Target : Integer; Value : Integer) : Integer
8841:  Function LockedExchangeAdd( var Target : Integer; Value : Integer) : Integer
8842:  Function LockedExchangeDec( var Target : Integer) : Integer
8843:  Function LockedExchangeInc( var Target : Integer) : Integer
8844:  Function LockedExchangeSub( var Target : Integer; Value : Integer) : Integer
8845:  Function LockedInc( var Target : Integer) : Integer
8846:  Function LockedSub( var Target : Integer; Value : Integer) : Integer
8847:   TJclWaitResult', '( wrAbandoned, wrError, wrIoCompletion, wrSignaled, wrTimeout )
8848:   SIRegister_TJclDispatcherObject(CL);
8849:  Function WaitForMultipleObjects(const Objects:array of
      TJclDispatcherObject;WaitAll:Bool;TimeOut:Cardinal):Cardinal;
8850:  Function WaitAlertableForMultipleObjects(const Objects : array of TJclDispatcherObject; WaitAll:Bool;
      TimeOut : Cardinal):Cardinal
8851:   SIRegister_TJclCriticalSection(CL);
8852:   SIRegister_TJclCriticalSectionEx(CL);
8853:   SIRegister_TJclEvent(CL);
8854:   SIRegister_TJclWaitableTimer(CL);
8855:   SIRegister_TJclSemaphore(CL);
8856:   SIRegister_TJclMutex(CL);
8857:   POptexSharedInfo', '^TOptexSharedInfo // will not work
8858:   TOptexSharedInfo', 'record SpinCount:Int; LockCount: Int; ThreadId:Longword; RecursionCount:Int; end
8859:   SIRegister_TJclOptex(CL);
8860:   TMrewPreferred', '( mpReaders, mpWriters, mpEqual )
8861:   TMrewThreadInfo', 'record ThreadId : Longword; RecursionCount: Integer; Reader : Boolean; end
8862:   TMrewThreadInfoArray', 'array of TMrewThreadInfo
8863:   SIRegister_TJclMultiReadExclusiveWrite(CL);
8864:   PMetSectSharedInfo', '^TMetSectSharedInfo // will not work
8865:   TMetSectSharedInfo', 'record Initialized : LongBool; SpinLock : '
8866:    +'Longint; ThreadsWaiting : Longint; AvailableCount : Longint; MaximumCount : Longint; end
8867:   PMeteredSection', '^TMeteredSection // will not work
8868:   TMeteredSection', 'record Event : THandle; FileMap : THandle; SharedInfo : PMetSectSharedInfo; end
8869:   SIRegister_TJclMeteredSection(CL);
8870:   TEventInfo', 'record EventType : Longint; Signaled : LongBool; end
8871:   TMutexInfo', 'record SignalState : Longint; Owned : Boolean; Abandoned : Boolean; end
8872:   TSemaphoreCounts', 'record CurrentCount : Longint; MaximumCount: Longint; end
8873:   TTimerInfo', 'record Remaining : TLargeInteger; Signaled : LongBool; end
8874:  Function QueryCriticalSection( CS : TJclCriticalSection; var Info : TRTLCriticalSection) : Boolean
8875:  Function QueryEvent( Handle : THandle; var Info : TEventInfo) : Boolean
8876:  Function QueryMutex( Handle : THandle; var Info : TMutexInfo) : Boolean
8877:  Function QuerySemaphore( Handle : THandle; var Info : TSemaphoreCounts) : Boolean
8878:  Function QueryTimer( Handle : THandle; var Info : TTimerInfo) : Boolean
8879:   FindClass('TOBJECT'),'EJclWin32HandleObjectError
8880:   FindClass('TOBJECT'),'EJclDispatcherObjectError
8881:   FindClass('TOBJECT'),'EJclCriticalSectionError
8882:   FindClass('TOBJECT'),'EJclEventError
8883:   FindClass('TOBJECT'),'EJclWaitableTimerError
8884:   FindClass('TOBJECT'),'EJclSemaphoreError
8885:   FindClass('TOBJECT'),'EJclMutexError
8886:   FindClass('TOBJECT'),'EJclMeteredSectionError
8887: end;
8888:
8889:
8890: //**************************unit uPSI_mORMotReport;
8891: Procedure SetCurrentPrinterAsDefault
8892: Function CurrentPrinterName : string
8893: Function mCurrentPrinterPaperSize : string
8894: Procedure UseDefaultPrinter
```

```
8895:
8896: procedure SIRegisterTSTREAM(Cl: TPSPascalCompiler);
8897: begin
8898:   with FindClass('TOBJECT'), 'TStream') do begin
8899:     IsAbstract := True;
8900:    //RegisterMethod('Function Read( var Buffer, Count : Longint) : Longint
8901:    // RegisterMethod('Function Write( const Buffer, Count : Longint) : Longint
8902:     function Read(Buffer:String;Count:LongInt):LongInt
8903:     function Write(Buffer:String;Count:LongInt):LongInt
8904:     procedure ReadAB(Buffer: TByteArray;Count:LongInt)
8905:     procedure WriteAB(Buffer: TByteArray;Count:LongInt)
8906:     procedure ReadABD(Buffer: TByteDynArray;Count:LongInt)
8907:     procedure WriteABD(Buffer: TByteDynArray;Count:LongInt)
8908:     procedure ReadAC(Buffer: TCharArray;Count:LongInt)
8909:     procedure WriteAC(Buffer: TCharArray;Count:LongInt)
8910:     procedure ReadACD(Buffer: TCharDynArray;Count:LongInt)
8911:     procedure WriteACD(Buffer: TCharDynArray;Count:LongInt)
8912:
8913:     function Seek(Offset:LongInt;Origin:Word):LongInt
8914:     procedure ReadBuffer(Buffer:String;Count:LongInt)
8915:     procedure WriteBuffer(Buffer:String;Count:LongInt)
8916:     procedure ReadBufferAB(Buffer: TByteArray;Count:LongInt)
8917:     procedure WriteBufferAB(Buffer: TByteArray;Count:LongInt)
8918:     procedure ReadBufferABD(Buffer: TByteDynArray;Count:LongInt)
8919:     procedure WriteBufferABD(Buffer: TByteDynArray;Count:LongInt)
8920:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
8921:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
8922:     procedure ReadBufferAWD(Buffer: TWordDynArray;Count:LongInt)
8923:     procedure WriteBufferAWD(Buffer: TWordDynArray;Count:LongInt)
8924:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
8925:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
8926:     procedure ReadBufferAC(Buffer: TCharArray;Count:LongInt)
8927:     procedure WriteBufferAC(Buffer: TCharArray;Count:LongInt)
8928:     procedure ReadBufferACD(Buffer: TCharDynArray;Count:LongInt)
8929:     procedure WriteBufferACD(Buffer: TCharDynArray;Count:LongInt)
8930:
8931:     procedure ReadBufferP(Buffer: PChar;Count:LongInt)
8932:     procedure WriteBufferP(Buffer: PChar;Count:LongInt)
8933:     function InstanceSize: Longint
8934:     Procedure FixupResourceHeader( FixupInfo : Integer)
8935:     Procedure ReadResHeader
8936:
8937:     {$IFDEF DELPHI4UP}
8938:     function CopyFrom(Source:TStream;Count:Int64):LongInt
8939:     {$ELSE}
8940:     function CopyFrom(Source:TStream;Count:Integer):LongInt
8941:     {$ENDIF}
8942:     RegisterProperty('Position', 'LongInt', iptrw);
8943:     RegisterProperty('Size', 'LongInt', iptrw);
8944:   end;
8945: end;
8946:
8947:
8948: { ***************************************************************
8949:   Unit DMATH - Interface for DMATH.DLL
8950:   *************************************************************** }
8951: // see more docs/dmath_manual.pdf
8952:
8953: Function InitEval : Integer
8954: Procedure SetVariable( VarName : Char; Value : Float)
8955: Procedure SetFunction( FuncName : String; Wrapper : TWrapper)
8956: Function Eval( ExpressionString : String) : Float
8957:
8958: unit dmath; //types are in built, others are external in DLL
8959: interface
8960: {$IFDEF DELPHI}
8961: uses
8962:   StdCtrls, Graphics;
8963: {$ENDIF}
8964: { ------------------------------------------------------------------
8965:   Types and constants
8966:   ------------------------------------------------------------------ }
8967: {$i types.inc}
8968: { ------------------------------------------------------------------
8969:   Error handling
8970:   ------------------------------------------------------------------ }
8971: procedure SetErrCode(ErrCode : Integer); external 'dmath';
8972: { Sets the error code }
8973: function DefaultVal(ErrCode : Integer; DefVal : Float) : Float; external 'dmath';
8974: { Sets error code and default function value }
8975: function MathErr : Integer; external 'dmath';
8976: { Returns the error code }
8977: { ------------------------------------------------------------------
8978:   Dynamic arrays
8979:   ------------------------------------------------------------------ }
8980: procedure SetAutoInit(AutoInit : Boolean); external 'dmath';
8981: { Sets the auto-initialization of arrays }
8982: procedure DimVector(var V : TVector; Ub : Integer); external 'dmath';
8983: { Creates floating point vector V[0..Ub] }
```

```
8984: procedure DimIntVector(var V : TIntVector; Ub : Integer); external 'dmath';
8985: { Creates integer vector V[0..Ub] }
8986: procedure DimCompVector(var V : TCompVector; Ub : Integer); external 'dmath';
8987: { Creates complex vector V[0..Ub] }
8988: procedure DimBoolVector(var V : TBoolVector; Ub : Integer); external 'dmath';
8989: { Creates boolean vector V[0..Ub] }
8990: procedure DimStrVector(var V : TStrVector; Ub : Integer); external 'dmath';
8991: { Creates string vector V[0..Ub] }
8992: procedure DimMatrix(var A : TMatrix; Ub1, Ub2 : Integer); external 'dmath';
8993: { Creates floating point matrix A[0..Ub1, 0..Ub2] }
8994: procedure DimIntMatrix(var A : TIntMatrix; Ub1, Ub2 : Integer); external 'dmath';
8995: { Creates integer matrix A[0..Ub1, 0..Ub2] }
8996: procedure DimCompMatrix(var A : TCompMatrix; Ub1, Ub2 : Integer); external 'dmath';
8997: { Creates complex matrix A[0..Ub1, 0..Ub2] }
8998: procedure DimBoolMatrix(var A : TBoolMatrix; Ub1, Ub2 : Integer); external 'dmath';
8999: { Creates boolean matrix A[0..Ub1, 0..Ub2] }
9000: procedure DimStrMatrix(var A : TStrMatrix; Ub1, Ub2 : Integer); external 'dmath';
9001: { Creates string matrix A[0..Ub1, 0..Ub2] }
9002: { ---------------------------------------------------------------
9003:   Minimum, maximum, sign and exchange
9004:   --------------------------------------------------------------- }
9005: function FMin(X, Y : Float) : Float; external 'dmath';
9006: { Minimum of 2 reals }
9007: function FMax(X, Y : Float) : Float; external 'dmath';
9008: { Maximum of 2 reals }
9009: function IMin(X, Y : Integer) : Integer; external 'dmath';
9010: { Minimum of 2 integers }
9011: function IMax(X, Y : Integer) : Integer; external 'dmath';
9012: { Maximum of 2 integers }
9013: function Sgn(X : Float) : Integer; external 'dmath';
9014: { Sign (returns 1 if X = 0) }
9015: function Sgn0(X : Float) : Integer; external 'dmath';
9016: { Sign (returns 0 if X = 0) }
9017: function DSgn(A, B : Float) : Float; external 'dmath';
9018: { Sgn(B) * |A| }
9019: procedure FSwap(var X, Y : Float); external 'dmath';
9020: { Exchange 2 reals }
9021: procedure ISwap(var X, Y : Integer); external 'dmath';
9022: { Exchange 2 integers }
9023: { ---------------------------------------------------------------
9024:   Rounding functions
9025:   --------------------------------------------------------------- }
9026: function RoundN(X : Float; N : Integer) : Float; external 'dmath';
9027: { Rounds X to N decimal places }
9028: function Ceil(X : Float) : Integer; external 'dmath';
9029: { Ceiling function }
9030: function Floor(X : Float) : Integer; external 'dmath';
9031: { Floor function }
9032: { ---------------------------------------------------------------
9033:   Logarithms, exponentials and power
9034:   --------------------------------------------------------------- }
9035: function Expo(X : Float) : Float; external 'dmath';
9036: { Exponential }
9037: function Exp2(X : Float) : Float; external 'dmath';
9038: { 2^X }
9039: function Exp10(X : Float) : Float; external 'dmath';
9040: { 10^X }
9041: function Log(X : Float) : Float; external 'dmath';
9042: { Natural log }
9043: function Log2(X : Float) : Float; external 'dmath';
9044: { Log, base 2 }
9045: function Log10(X : Float) : Float; external 'dmath';
9046: { Decimal log }
9047: function LogA(X, A : Float) : Float; external 'dmath';
9048: { Log, base A }
9049: function IntPower(X : Float; N : Integer) : Float; external 'dmath';
9050: { X^N }
9051: function Power(X, Y : Float) : Float; external 'dmath';
9052: { X^Y, X >= 0 }
9053: { ---------------------------------------------------------------
9054:   Trigonometric functions
9055:   --------------------------------------------------------------- }
9056: function Pythag(X, Y : Float) : Float; external 'dmath';
9057: { Sqrt(X^2 + Y^2) }
9058: function FixAngle(Theta : Float) : Float; external 'dmath';
9059: { Set Theta in -Pi..Pi }
9060: function Tan(X : Float) : Float; external 'dmath';
9061: { Tangent }
9062: function ArcSin(X : Float) : Float; external 'dmath';
9063: { Arc sinus }
9064: function ArcCos(X : Float) : Float; external 'dmath';
9065: { Arc cosinus }
9066: function ArcTan2(Y, X : Float) : Float; external 'dmath';
9067: { Angle (Ox, OM) with M(X,Y) }
9068: { ---------------------------------------------------------------
9069:   Hyperbolic functions
9070:   --------------------------------------------------------------- }
9071: function Sinh(X : Float) : Float; external 'dmath';
9072: { Hyperbolic sine }
```

```
9073: function Cosh(X : Float) : Float; external 'dmath';
9074: { Hyperbolic cosine }
9075: function Tanh(X : Float) : Float; external 'dmath';
9076: { Hyperbolic tangent }
9077: function ArcSinh(X : Float) : Float; external 'dmath';
9078: { Inverse hyperbolic sine }
9079: function ArcCosh(X : Float) : Float; external 'dmath';
9080: { Inverse hyperbolic cosine }
9081: function ArcTanh(X : Float) : Float; external 'dmath';
9082: { Inverse hyperbolic tangent }
9083: procedure SinhCosh(X : Float; var SinhX, CoshX : Float); external 'dmath';
9084: { Sinh & Cosh }
9085: { ---------------------------------------------------------------
9086:   Gamma function and related functions
9087:   --------------------------------------------------------------- }
9088: function Fact(N : Integer) : Float; external 'dmath';
9089: { Factorial }
9090: function SgnGamma(X : Float) : Integer; external 'dmath';
9091: { Sign of Gamma function }
9092: function Gamma(X : Float) : Float; external 'dmath';
9093: { Gamma function }
9094: function LnGamma(X : Float) : Float; external 'dmath';
9095: { Logarithm of Gamma function }
9096: function Stirling(X : Float) : Float; external 'dmath';
9097: { Stirling's formula for the Gamma function }
9098: function StirLog(X : Float) : Float; external 'dmath';
9099: { Approximate Ln(Gamma) by Stirling's formula, for X >= 13 }
9100: function DiGamma(X : Float ) : Float; external 'dmath';
9101: { Digamma function }
9102: function TriGamma(X : Float ) : Float; external 'dmath';
9103: { Trigamma function }
9104: function IGamma(A, X : Float) : Float; external 'dmath';
9105: { Incomplete Gamma function}
9106: function JGamma(A, X : Float) : Float; external 'dmath';
9107: { Complement of incomplete Gamma function }
9108: function InvGamma(A, P : Float) : Float; external 'dmath';
9109: { Inverse of incomplete Gamma function }
9110: function Erf(X : Float) : Float; external 'dmath';
9111: { Error function }
9112: function Erfc(X : Float) : Float; external 'dmath';
9113: { Complement of error function }
9114: { ---------------------------------------------------------------
9115:   Beta function and related functions
9116:   --------------------------------------------------------------- }
9117: function Beta(X, Y : Float) : Float; external 'dmath';
9118: { Beta function }
9119: function IBeta(A, B, X : Float) : Float; external 'dmath';
9120: { Incomplete Beta function }
9121: function InvBeta(A, B, Y : Float) : Float; external 'dmath';
9122: { Inverse of incomplete Beta function }
9123: { ---------------------------------------------------------------
9124:   Lambert's function
9125:   --------------------------------------------------------------- }
9126: function LambertW(X : Float; UBranch, Offset : Boolean) : Float; external 'dmath';
9127:  ---------------------------------------------------------------
9128:   Binomial distribution
9129:   --------------------------------------------------------------- }
9130: function Binomial(N, K : Integer) : Float; external 'dmath';
9131: { Binomial coefficient C(N,K) }
9132: function PBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9133: { Probability of binomial distribution }
9134: function FBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9135: { Cumulative probability for binomial distrib. }
9136: { ---------------------------------------------------------------
9137:   Poisson distribution
9138:   --------------------------------------------------------------- }
9139: function PPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9140: { Probability of Poisson distribution }
9141: function FPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9142: { Cumulative probability for Poisson distrib. }
9143: { ---------------------------------------------------------------
9144:   Exponential distribution
9145:   --------------------------------------------------------------- }
9146: function DExpo(A, X : Float) : Float; external 'dmath';
9147: { Density of exponential distribution with parameter A }
9148: function FExpo(A, X : Float) : Float; external 'dmath';
9149: { Cumulative probability function for exponential dist. with parameter A }
9150: { ---------------------------------------------------------------
9151:   Standard normal distribution
9152:   --------------------------------------------------------------- }
9153: function DNorm(X : Float) : Float; external 'dmath';
9154: { Density of standard normal distribution }
9155: function FNorm(X : Float) : Float; external 'dmath';
9156: { Cumulative probability for standard normal distrib. }
9157: function PNorm(X : Float) : Float; external 'dmath';
9158: { Prob(|U| > X) for standard normal distrib. }
9159: function InvNorm(P : Float) : Float; external 'dmath';
9160: { Inverse of standard normal distribution }
9161: { ---------------------------------------------------------------
```

```
9162:   Student's distribution
9163:   ------------------------------------------------------------------ }
9164: function DStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9165: { Density of Student distribution with Nu d.o.f. }
9166: function FStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9167: { Cumulative probability for Student distrib. with Nu d.o.f. }
9168: function PStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9169: { Prob(|t| > X) for Student distrib. with Nu d.o.f. }
9170: function InvStudent(Nu : Integer; P : Float) : Float; external 'dmath';
9171: { Inverse of Student's t-distribution function }
9172: { ------------------------------------------------------------------
9173:   Khi-2 distribution
9174:   ------------------------------------------------------------------ }
9175: function DKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9176: { Density of Khi-2 distribution with Nu d.o.f. }
9177: function FKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9178: { Cumulative prob. for Khi-2 distrib. with Nu d.o.f. }
9179: function PKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9180: { Prob(Khi2 > X) for Khi-2 distrib. with Nu d.o.f. }
9181: function InvKhi2(Nu : Integer; P : Float) : Float; external 'dmath';
9182: { Inverse of Khi-2 distribution function }
9183: { ------------------------------------------------------------------
9184:   Fisher-Snedecor distribution
9185:   ------------------------------------------------------------------ }
9186: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9187: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
9188: function FSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9189: { Cumulative prob. for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9190: function PSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9191: { Prob(F > X) for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9192: function InvSnedecor(Nu1, Nu2 : Integer; P : Float) : Float; external 'dmath';
9193: { Inverse of Snedecor's F-distribution function }
9194: { ------------------------------------------------------------------
9195:   Beta distribution
9196:   ------------------------------------------------------------------ }
9197: function DBeta(A, B, X : Float) : Float; external 'dmath';
9198: { Density of Beta distribution with parameters A and B }
9199: function FBeta(A, B, X : Float) : Float; external 'dmath';
9200: { Cumulative probability for Beta distrib. with param. A and B }
9201: { ------------------------------------------------------------------
9202:   Gamma distribution
9203:   ------------------------------------------------------------------ }
9204: function DGamma(A, B, X : Float) : Float; external 'dmath';
9205: { Density of Gamma distribution with parameters A and B }
9206: function FGamma(A, B, X : Float) : Float; external 'dmath';
9207: { Cumulative probability for Gamma distrib. with param. A and B }
9208: { ------------------------------------------------------------------
9209:   Expression evaluation
9210:   ------------------------------------------------------------------ }
9211: function InitEval : Integer; external 'dmath';
9212: { Initializes built-in functions and returns their number }
9213: function Eval(ExpressionString : String) : Float; external 'dmath';
9214: { Evaluates an expression at run-time }
9215: procedure SetVariable(VarName : Char; Value : Float); external 'dmath';
9216: { Assigns a value to a variable }
9217: procedure SetFunction(FuncName : String; Wrapper : TWrapper); external 'dmath';
9218: { Adds a function to the parser }
9219: { ------------------------------------------------------------------
9220:   Matrices and linear equations
9221:   ------------------------------------------------------------------ }
9222: procedure GaussJordan(A           : TMatrix;
9223:                       Lb, Ub1, Ub2 : Integer;
9224:                       var Det      : Float); external 'dmath';
9225: { Transforms a matrix according to the Gauss-Jordan method }
9226: procedure LinEq(A       : TMatrix;
9227:                 B       : TVector;
9228:                 Lb, Ub  : Integer;
9229:                 var Det : Float); external 'dmath';
9230: { Solves a linear system according to the Gauss-Jordan method }
9231: procedure Cholesky(A, L : TMatrix; Lb, Ub : Integer); external 'dmath';
9232: { Cholesky factorization of a positive definite symmetric matrix }
9233: procedure LU_Decomp(A : TMatrix; Lb, Ub : Integer); external 'dmath';
9234: { LU decomposition }
9235: procedure LU_Solve(A      : TMatrix;
9236:                    B      : TVector;
9237:                    Lb, Ub : Integer;
9238:                    X      : TVector); external 'dmath';
9239: { Solution of linear system from LU decomposition }
9240: procedure QR_Decomp(A           : TMatrix;
9241:                     Lb, Ub1, Ub2 : Integer;
9242:                     R           : TMatrix); external 'dmath';
9243: { QR decomposition }
9244: procedure QR_Solve(Q, R         : TMatrix;
9245:                    B           : TVector;
9246:                    Lb, Ub1, Ub2 : Integer;
9247:                    X           : TVector); external 'dmath';
9248: { Solution of linear system from QR decomposition }
9249: procedure SV_Decomp(A           : TMatrix;
9250:                     Lb, Ub1, Ub2 : Integer;
```

```
                    S            : TVector;
                    V            : TMatrix); external 'dmath';
{ Singular value decomposition }
procedure SV_SetZero(S      : TVector;
                     Lb, Ub : Integer;
                     Tol    : Float); external 'dmath';
{ Set lowest singular values to zero }
procedure SV_Solve(U            : TMatrix;
                   S            : TVector;
                   V            : TMatrix;
                   B            : TVector;
                   Lb, Ub1, Ub2 : Integer;
                   X            : TVector); external 'dmath';
{ Solution of linear system from SVD }
procedure SV_Approx(U            : TMatrix;
                    S            : TVector;
                    V            : TMatrix;
                    Lb, Ub1, Ub2 : Integer;
                    A            : TMatrix); external 'dmath';
{ Matrix approximation from SVD }
procedure EigenVals(A      : TMatrix;
                    Lb, Ub : Integer;
                    Lambda : TCompVector); external 'dmath';
{ Eigenvalues of a general square matrix }
procedure EigenVect(A      : TMatrix;
                    Lb, Ub : Integer;
                    Lambda : TCompVector;
                    V      : TMatrix); external 'dmath';
{ Eigenvalues and eigenvectors of a general square matrix }
procedure EigenSym(A      : TMatrix;
                   Lb, Ub : Integer;
                   Lambda : TVector;
                   V      : TMatrix); external 'dmath';
{ Eigenvalues and eigenvectors of a symmetric matrix (SVD method) }
procedure Jacobi(A               : TMatrix;
                 Lb, Ub, MaxIter : Integer;
                 Tol             : Float;
                 Lambda          : TVector;
                 V               : TMatrix); external 'dmath';
{ Eigenvalues and eigenvectors of a symmetric matrix (Jacobi method) }
{ ---------------------------------------------------------------
  Optimization
  --------------------------------------------------------------- }
procedure MinBrack(Func                    : TFunc;
                   var A, B, C, Fa, Fb, Fc : Float); external 'dmath';
{ Brackets a minimum of a function }
procedure GoldSearch(Func           : TFunc;
                     A, B           : Float;
                     MaxIter        : Integer;
                     Tol            : Float;
                     var Xmin, Ymin : Float); external 'dmath';
{ Minimization of a function of one variable (golden search) }
procedure LinMin(Func      : TFuncNVar;
                 X, DeltaX : TVector;
                 Lb, Ub    : Integer;
                 var R     : Float;
                 MaxIter   : Integer;
                 Tol       : Float;
                 var F_min : Float); external 'dmath';
{ Minimization of a function of several variables along a line }
procedure Newton(Func      : TFuncNVar;
                 HessGrad  : THessGrad;
                 X         : TVector;
                 Lb, Ub    : Integer;
                 MaxIter   : Integer;
                 Tol       : Float;
                 var F_min : Float;
                 G         : TVector;
                 H_inv     : TMatrix;
                 var Det   : Float); external 'dmath';
{ Minimization of a function of several variables (Newton's method) }
procedure SaveNewton(FileName : string); external 'dmath';
{ Save Newton iterations in a file }
procedure Marquardt(Func      : TFuncNVar;
                    HessGrad  : THessGrad;
                    X         : TVector;
                    Lb, Ub    : Integer;
                    MaxIter   : Integer;
                    Tol       : Float;
                    var F_min : Float;
                    G         : TVector;
                    H_inv     : TMatrix;
                    var Det   : Float); external 'dmath';
{ Minimization of a function of several variables (Marquardt's method) }
procedure SaveMarquardt(FileName : string); external 'dmath';
{ Save Marquardt iterations in a file }
procedure BFGS(Func      : TFuncNVar;
               Gradient  : TGradient;
               X         : TVector;
```

```
9340:               Lb, Ub    : Integer;
9341:               MaxIter   : Integer;
9342:               Tol       : Float;
9343:               var F_min : Float;
9344:               G         : TVector;
9345:               H_inv     : TMatrix); external 'dmath';
9346: { Minimization of a function of several variables (BFGS method) }
9347: procedure SaveBFGS(FileName : string); external 'dmath';
9348: { Save BFGS iterations in a file }
9349: procedure Simplex(Func      : TFuncNVar;
9350:                   X         : TVector;
9351:                   Lb, Ub    : Integer;
9352:                   MaxIter   : Integer;
9353:                   Tol       : Float;
9354:                   var F_min : Float); external 'dmath';
9355: { Minimization of a function of several variables (Simplex) }
9356: procedure SaveSimplex(FileName : string); external 'dmath';
9357: { Save Simplex iterations in a file }
9358: { ---------------------------------------------------------------
9359:   Nonlinear equations
9360:   --------------------------------------------------------------- }
9361: procedure RootBrack(Func            : TFunc;
9362:                     var X, Y, FX, FY : Float); external 'dmath';
9363: { Brackets a root of function Func between X and Y }
9364: procedure Bisect(Func     : TFunc;
9365:                  var X, Y : Float;
9366:                  MaxIter  : Integer;
9367:                  Tol      : Float;
9368:                  var F    : Float); external 'dmath';
9369: { Bisection method }
9370: procedure Secant(Func     : TFunc;
9371:                  var X, Y : Float;
9372:                  MaxIter  : Integer;
9373:                  Tol      : Float;
9374:                  var F    : Float); external 'dmath';
9375: { Secant method }
9376: procedure NewtEq(Func, Deriv : TFunc;
9377:                  var X       : Float;
9378:                  MaxIter     : Integer;
9379:                  Tol         : Float;
9380:                  var F       : Float); external 'dmath';
9381: { Newton-Raphson method for a single nonlinear equation }
9382: procedure NewtEqs(Equations : TEquations;
9383:                   Jacobian  : TJacobian;
9384:                   X, F      : TVector;
9385:                   Lb, Ub    : Integer;
9386:                   MaxIter   : Integer;
9387:                   Tol       : Float); external 'dmath';
9388: { Newton-Raphson method for a system of nonlinear equations }
9389: procedure Broyden(Equations : TEquations;
9390:                   X, F      : TVector;
9391:                   Lb, Ub    : Integer;
9392:                   MaxIter   : Integer;
9393:                   Tol       : Float); external 'dmath';
9394: { Broyden's method for a system of nonlinear equations }
9395: { ---------------------------------------------------------------
9396:   Polynomials and rational fractions
9397:   --------------------------------------------------------------- }
9398: function Poly(X    : Float;
9399:               Coef : TVector;
9400:               Deg  : Integer) : Float; external 'dmath';
9401: { Evaluates a polynomial }
9402: function RFrac(X          : Float;
9403:                Coef       : TVector;
9404:                Deg1, Deg2 : Integer) : Float; external 'dmath';
9405: { Evaluates a rational fraction }
9406: function RootPol1(A, B  : Float;
9407:                   var X : Float) : Integer; external 'dmath';
9408: { Solves the linear equation A + B * X = 0 }
9409: function RootPol2(Coef : TVector;
9410:                   Z    : TCompVector) : Integer; external 'dmath';
9411: { Solves a quadratic equation }
9412: function RootPol3(Coef : TVector;
9413:                   Z    : TCompVector) : Integer; external 'dmath';
9414: { Solves a cubic equation }
9415: function RootPol4(Coef : TVector;
9416:                   Z    : TCompVector) : Integer; external 'dmath';
9417: { Solves a quartic equation }
9418: function RootPol(Coef : TVector;
9419:                  Deg  : Integer;
9420:                  Z    : TCompVector) : Integer; external 'dmath';
9421: { Solves a polynomial equation }
9422: function SetRealRoots(Deg : Integer;
9423:                       Z   : TCompVector;
9424:                       Tol : Float) : Integer; external 'dmath';
9425: { Set the imaginary part of a root to zero }
9426: procedure SortRoots(Deg : Integer;
9427:                     Z   : TCompVector); external 'dmath';
9428: { Sorts the roots of a polynomial }
```

```
9429: { ------------------------------------------------------------------
9430:   Numerical integration and differential equations
9431:   ------------------------------------------------------------------ }
9432: function TrapInt(X, Y : TVector; N : Integer) : Float; external 'dmath';
9433: { Integration by trapezoidal rule }
9434: function GausLeg(Func : TFunc; A, B : Float) : Float; external 'dmath';
9435: { Integral from A to B }
9436: function GausLeg0(Func : TFunc; B : Float) : Float; external 'dmath';
9437: { Integral from 0 to B }
9438: function Convol(Func1, Func2 : TFunc; T : Float) : Float; external 'dmath';
9439: { Convolution product at time T }
9440: procedure ConvTrap(Func1,Func2:TFunc; T,Y:TVector; N:Integer);external 'dmath';
9441: { Convolution by trapezoidal rule }
9442: procedure RKF45(F                    : TDiffEqs;
9443:                 Neqn                 : Integer;
9444:                 Y, Yp                : TVector;
9445:                 var T                : Float;
9446:                 Tout, RelErr, AbsErr : Float;
9447:                 var Flag             : Integer); external 'dmath';
9448: { Integration of a system of differential equations }
9449: { ------------------------------------------------------------------
9450:   Fast Fourier Transform
9451:   ------------------------------------------------------------------ }
9452: procedure FFT(NumSamples        : Integer;
9453:               InArray, OutArray : TCompVector); external 'dmath';
9454: { Fast Fourier Transform }
9455: procedure IFFT(NumSamples        : Integer;
9456:                InArray, OutArray : TCompVector); external 'dmath';
9457: { Inverse Fast Fourier Transform }
9458: procedure FFT_Integer(NumSamples     : Integer;
9459:                       RealIn, ImagIn : TIntVector;
9460:                       OutArray       : TCompVector); external 'dmath';
9461: { Fast Fourier Transform for integer data }
9462: procedure FFT_Integer_Cleanup; external 'dmath';
9463: { Clear memory after a call to FFT_Integer }
9464: procedure CalcFrequency(NumSamples,
9465:                         FrequencyIndex : Integer;
9466:                         InArray        : TCompVector;
9467:                         var FFT        : Complex); external 'dmath';
9468: { Direct computation of Fourier transform }
9469: { ------------------------------------------------------------------
9470:   Random numbers
9471:   ------------------------------------------------------------------ }
9472: procedure SetRNG(RNG : RNG_Type); external 'dmath';
9473: { Select generator }
9474: procedure InitGen(Seed : RNG_IntType); external 'dmath';
9475: { Initialize generator }
9476: function IRanGen : RNG_IntType; external 'dmath';
9477: { 32-bit random integer in [-2^31 .. 2^31 - 1] }
9478: function IRanGen31 : RNG_IntType; external 'dmath';
9479: { 31-bit random integer in [0 .. 2^31 - 1] }
9480: function RanGen1 : Float; external 'dmath';
9481: { 32-bit random real in [0,1] }
9482: function RanGen2 : Float; external 'dmath';
9483: { 32-bit random real in [0,1) }
9484: function RanGen3 : Float; external 'dmath';
9485: { 32-bit random real in (0,1) }
9486: function RanGen53 : Float; external 'dmath';
9487: { 53-bit random real in [0,1) }
9488: procedure InitMWC(Seed : RNG_IntType); external 'dmath';
9489: { Initializes the 'Multiply with carry' random number generator }
9490: function IRanMWC : RNG_IntType; external 'dmath';
9491: { Returns a 32 bit random number in [-2^31 ; 2^31-1] }
9492: procedure InitMT(Seed : RNG_IntType); external 'dmath';
9493: { Initializes Mersenne Twister generator with a seed }
9494: procedure InitMTbyArray(InitKey   : array of RNG_LongType;
9495:                         KeyLength : Word); external 'dmath';
9496: { Initialize MT generator with an array InitKey[0..(KeyLength - 1)] }
9497: function IRanMT : RNG_IntType; external 'dmath';
9498: { Random integer from MT generator }
9499: procedure InitUVAGbyString(KeyPhrase : string); external 'dmath';
9500: { Initializes the UVAG generator with a string }
9501: procedure InitUVAG(Seed : RNG_IntType); external 'dmath';
9502: { Initializes the UVAG generator with an integer }
9503: function IRanUVAG : RNG_IntType; external 'dmath';
9504: { Random integer from UVAG generator }
9505: function RanGaussStd : Float; external 'dmath';
9506: { Random number from standard normal distribution }
9507: function RanGauss(Mu, Sigma : Float) : Float; external 'dmath';
9508: { Random number from normal distrib. with mean Mu and S. D. Sigma }
9509: procedure RanMult(M      : TVector; L      : TMatrix;
9510:                   Lb, Ub : Integer;
9511:                   X      : TVector); external 'dmath';
9512: { Random vector from multinormal distribution (correlated) }
9513: procedure RanMultIndep(M, S   : TVector;
9514:                        Lb, Ub : Integer;
9515:                        X      : TVector); external 'dmath';
9516: { Random vector from multinormal distribution (uncorrelated) }
9517: procedure InitMHParams(NCycles, MaxSim, SavedSim : Integer); external 'dmath';
```

```
9518: { Initializes Metropolis-Hastings parameters }
9519: procedure GetMHParams(var NCycles, MaxSim,SavedSim:Integer); external 'dmath';
9520: { Returns Metropolis-Hastings parameters }
9521: procedure Hastings(Func      : TFuncNVar;
9522:                    T         : Float;
9523:                    X         : TVector;
9524:                    V         : TMatrix;
9525:                    Lb, Ub    : Integer;
9526:                    Xmat      : TMatrix;
9527:                    X_min     : TVector;
9528:                    var F_min : Float); external 'dmath';
9529: { Simulation of a probability density function by Metropolis-Hastings }
9530: procedure InitSAParams(NT, NS, NCycles : Integer; RT : Float); external 'dmath';
9531: { Initializes Simulated Annealing parameters }
9532: procedure SA_CreateLogFile(FileName : String); external 'dmath';
9533: { Initializes log file }
9534: procedure SimAnn(Func          : TFuncNVar;
9535:                  X, Xmin, Xmax : TVector;
9536:                  Lb, Ub        : Integer;
9537:                  var F_min     : Float); external 'dmath';
9538: { Minimization of a function of several var. by simulated annealing }
9539: procedure InitGAParams(NP, NG : Integer; SR, MR, HR : Float); external 'dmath';
9540: { Initializes Genetic Algorithm parameters }
9541: procedure GA_CreateLogFile(FileName : String); external 'dmath';
9542: { Initializes log file }
9543: procedure GenAlg(Func          : TFuncNVar;
9544:                  X, Xmin, Xmax : TVector;
9545:                  Lb, Ub        : Integer;
9546:                  var F_min     : Float); external 'dmath';
9547: { Minimization of a function of several var. by genetic algorithm }
9548: { ---------------------------------------------------------------------
9549:   Statistics
9550:   --------------------------------------------------------------------- }
9551: function Mean(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9552: { Mean of sample X }
9553: function Min(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9554: { Minimum of sample X }
9555: function Max(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9556: { Maximum of sample X }
9557: function Median(X : TVector; Lb, Ub : Integer; Sorted : Boolean) : Float; external 'dmath';
9558: { Median of sample X }
9559: function StDev(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9560: { Standard deviation estimated from sample X }
9561: function StDevP(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9562: { Standard deviation of population }
9563: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9564: { Correlation coefficient }
9565: function Skewness(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9566: { Skewness of sample X }
9567: function Kurtosis(X : TVector; Lb, Ub : Integer; M,Sigma: Float): Float; external 'dmath';
9568: { Kurtosis of sample X }
9569: procedure QSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9570: { Quick sort (ascending order) }
9571: procedure DQSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9572: { Quick sort (descending order) }
9573: procedure Interval(X1, X2             : Float;
9574:                    MinDiv, MaxDiv     : Integer;
9575:                    var Min, Max, Step : Float); external 'dmath';
9576: { Determines an interval for a set of values }
9577: procedure AutoScale(X : TVector; Lb, Ub : Integer; Scale : TScale;
9578:                     var XMin, XMax, XStep : Float); external 'dmath';
9579: { Finds an appropriate scale for plotting the data in X[Lb..Ub] }
9580: procedure StudIndep(N1, N2          : Integer;
9581:                     M1, M2, S1, S2 : Float;
9582:                     var T          : Float;
9583:                     var DoF        : Integer); external 'dmath';
9584: { Student t-test for independent samples }
9585: procedure StudPaired(X, Y    : TVector;
9586:                      Lb, Ub  : Integer;
9587:                      var T   : Float;
9588:                      var DoF : Integer); external 'dmath';
9589: { Student t-test for paired samples }
9590: procedure AnOVa1(Ns               : Integer;
9591:                  N                : TIntVector;
9592:                  M, S             : TVector;
9593:                  var V_f, V_r, F  : Float;
9594:                  var DoF_f, DoF_r : Integer); external 'dmath';
9595: { One-way analysis of variance }
9596: procedure AnOVa2(NA, NB, Nobs : Integer;
9597:                  M, S         : TMatrix;
9598:                  V, F         : TVector;
9599:                  DoF          : TIntVector); external 'dmath';
9600: { Two-way analysis of variance }
9601: procedure Snedecor(N1, N2          : Integer;
9602:                    S1, S2          : Float;
9603:                    var F           : Float;
9604:                    var DoF1, DoF2 : Integer); external 'dmath';
9605: { Snedecor's F-test (comparison of two variances) }
9606: procedure Bartlett(Ns       : Integer;
```

```
9607:                   N        : TIntVector;
9608:                   S        : TVector;
9609:                   var Khi2 : Float;
9610:                   var DoF  : Integer); external 'dmath';
9611: { Bartlett's test (comparison of several variances) }
9612: procedure Mann_Whitney(N1, N2     : Integer;
9613:                        X1, X2     : TVector;
9614:                        var U, Eps : Float); external 'dmath';
9615: { Mann-Whitney test}
9616: procedure Wilcoxon(X, Y       : TVector;
9617:                   Lb, Ub     : Integer;
9618:                   var Ndiff  : Integer;
9619:                   var T, Eps : Float); external 'dmath';
9620: { Wilcoxon test }
9621: procedure Kruskal_Wallis(Ns      : Integer;
9622:                          N       : TIntVector;
9623:                          X       : TMatrix;
9624:                          var H   : Float;
9625:                          var DoF : Integer); external 'dmath';
9626: { Kruskal-Wallis test }
9627: procedure Khi2_Conform(N_cls    : Integer;
9628:                        N_estim  : Integer;
9629:                        Obs      : TIntVector;
9630:                        Calc     : TVector;
9631:                        var Khi2 : Float;
9632:                        var DoF  : Integer); external 'dmath';
9633: { Khi-2 test for conformity }
9634: procedure Khi2_Indep(N_lin    : Integer;
9635:                      N_col    : Integer;
9636:                      Obs      : TIntMatrix;
9637:                      var Khi2 : Float;
9638:                      var DoF  : Integer); external 'dmath';
9639: { Khi-2 test for independence }
9640: procedure Woolf_Conform(N_cls   : Integer;
9641:                         N_estim : Integer;
9642:                         Obs     : TIntVector;
9643:                         Calc    : TVector;
9644:                         var G   : Float;
9645:                         var DoF : Integer); external 'dmath';
9646: { Woolf's test for conformity }
9647: procedure Woolf_Indep(N_lin   : Integer;
9648:                       N_col   : Integer;
9649:                       Obs     : TIntMatrix;
9650:                       var G   : Float;
9651:                       var DoF : Integer); external 'dmath';
9652: { Woolf's test for independence }
9653: procedure DimStatClassVector(var C : TStatClassVector;
9654:                              Ub    : Integer); external 'dmath';
9655: { Allocates an array of statistical classes: C[0..Ub] }
9656: procedure Distrib(X       : TVector;
9657:                   Lb, Ub  : Integer;
9658:                   A, B, H : Float;
9659:                   C       : TStatClassVector); external 'dmath';
9660: { Distributes an array X[Lb..Ub] into statistical classes }
9661: { ----------------------------------------------------------------
9662:   Linear / polynomial regression
9663:   ---------------------------------------------------------------- }
9664: procedure LinFit(X, Y   : TVector;
9665:                  Lb, Ub : Integer;
9666:                  B      : TVector;
9667:                  V      : TMatrix); external 'dmath';
9668: { Linear regression : Y = B(0) + B(1) * X }
9669: procedure WLinFit(X, Y, S : TVector;
9670:                   Lb, Ub  : Integer;
9671:                   B       : TVector;
9672:                   V       : TMatrix); external 'dmath';
9673: { Weighted linear regression : Y = B(0) + B(1) * X }
9674: procedure SVDLinFit(X, Y   : TVector;
9675:                     Lb, Ub : Integer;
9676:                     SVDTol : Float;
9677:                     B      : TVector;
9678:                     V      : TMatrix); external 'dmath';
9679: { Unweighted linear regression by singular value decomposition }
9680: procedure WSVDLinFit(X, Y, S : TVector;
9681:                      Lb, Ub  : Integer;
9682:                      SVDTol  : Float;
9683:                      B       : TVector;
9684:                      V       : TMatrix); external 'dmath';
9685: { Weighted linear regression by singular value decomposition }
9686: procedure MulFit(X            : TMatrix;
9687:                  Y            : TVector;
9688:                  Lb, Ub, Nvar : Integer;
9689:                  ConsTerm     : Boolean;
9690:                  B            : TVector;
9691:                  V            : TMatrix); external 'dmath';
9692: { Multiple linear regression by Gauss-Jordan method }
9693: procedure WMulFit(X           : TMatrix;
9694:                   Y, S        : TVector;
9695:                   Lb, Ub, Nvar : Integer;
```

```
9696:                  ConsTerm     : Boolean;
9697:                  B            : TVector;
9698:                  V            : TMatrix); external 'dmath';
9699: { Weighted multiple linear regression by Gauss-Jordan method }
9700: procedure SVDFit(X            : TMatrix;
9701:                  Y            : TVector;
9702:                 Lb, Ub, Nvar : Integer;
9703:                 ConsTerm     : Boolean;
9704:                 SVDTol       : Float;
9705:                 B            : TVector;
9706:                 V            : TMatrix); external 'dmath';
9707: { Multiple linear regression by singular value decomposition }
9708: procedure WSVDFit(X            : TMatrix;
9709:                  Y, S         : TVector;
9710:                  Lb, Ub, Nvar : Integer;
9711:                  ConsTerm     : Boolean;
9712:                  SVDTol       : Float;
9713:                  B            : TVector;
9714:                  V            : TMatrix); external 'dmath';
9715: { Weighted multiple linear regression by singular value decomposition }
9716: procedure PolFit(X, Y        : TVector;
9717:                 Lb, Ub, Deg : Integer;
9718:                 B           : TVector;
9719:                 V           : TMatrix); external 'dmath';
9720: { Polynomial regression by Gauss-Jordan method }
9721: procedure WPolFit(X, Y, S     : TVector;
9722:                  Lb, Ub, Deg : Integer;
9723:                  B           : TVector;
9724:                  V           : TMatrix); external 'dmath';
9725: { Weighted polynomial regression by Gauss-Jordan method }
9726: procedure SVDPolFit(X, Y        : TVector;
9727:                    Lb, Ub, Deg : Integer;
9728:                    SVDTol      : Float;
9729:                    B           : TVector;
9730:                    V           : TMatrix); external 'dmath';
9731: { Unweighted polynomial regression by singular value decomposition }
9732: procedure WSVDPolFit(X, Y, S     : TVector;
9733:                     Lb, Ub, Deg : Integer;
9734:                     SVDTol      : Float;
9735:                     B           : TVector;
9736:                     V           : TMatrix); external 'dmath';
9737: { Weighted polynomial regression by singular value decomposition }
9738: procedure RegTest(Y, Ycalc : TVector;
9739:                  LbY, UbY : Integer;
9740:                  V        : TMatrix;
9741:                  LbV, UbV : Integer;
9742:                  var Test : TRegTest); external 'dmath';
9743: { Test of unweighted regression }
9744: procedure WRegTest(Y, Ycalc, S : TVector;
9745:                   LbY, UbY    : Integer;
9746:                   V           : TMatrix;
9747:                   LbV, UbV    : Integer;
9748:                   var Test    : TRegTest); external 'dmath';
9749: { Test of weighted regression }
9750: { ------------------------------------------------------------------
9751:   Nonlinear regression
9752:   ------------------------------------------------------------------ }
9753: procedure SetOptAlgo(Algo : TOptAlgo); external 'dmath';
9754: { Sets the optimization algorithm for nonlinear regression }
9755: function GetOptAlgo : TOptAlgo; external 'dmath';
9756: { Returns the optimization algorithm }
9757: procedure SetMaxParam(N : Byte); external 'dmath';
9758: { Sets the maximum number of regression parameters for nonlinear regression }
9759: function GetMaxParam : Byte; external 'dmath';
9760: { Returns the maximum number of regression parameters for nonlinear regression }
9761: procedure SetParamBounds(I : Byte; ParamMin, ParamMax : Float); external 'dmath';
9762: { Sets the bounds on the I-th regression parameter }
9763: procedure GetParamBounds(I : Byte; var ParamMin,ParamMax:Float); external 'dmath';
9764: { Returns the bounds on the I-th regression parameter }
9765: procedure NLFit(RegFunc   : TRegFunc;
9766:                DerivProc : TDerivProc;
9767:                X, Y      : TVector;
9768:                Lb, Ub    : Integer;
9769:                MaxIter   : Integer;
9770:                Tol       : Float;
9771:                B         : TVector;
9772:                FirstPar,
9773:                LastPar   : Integer;
9774:                V         : TMatrix); external 'dmath';
9775: { Unweighted nonlinear regression }
9776: procedure WNLFit(RegFunc   : TRegFunc;
9777:                 DerivProc : TDerivProc;
9778:                 X, Y, S   : TVector;
9779:                 Lb, Ub    : Integer;
9780:                 MaxIter   : Integer;
9781:                 Tol       : Float;
9782:                 B         : TVector;
9783:                 FirstPar,
9784:                 LastPar   : Integer;
```

```
9785:                  V         : TMatrix); external 'dmath';
9786: { Weighted nonlinear regression }
9787: procedure SetMCFile(FileName : String); external 'dmath';
9788: { Set file for saving MCMC simulations }
9789: procedure SimFit(RegFunc   : TRegFunc;
9790:                  X, Y      : TVector;
9791:                  Lb, Ub    : Integer;
9792:                  B         : TVector;
9793:                  FirstPar,
9794:                  LastPar   : Integer;
9795:                  V         : TMatrix); external 'dmath';
9796: { Simulation of unweighted nonlinear regression by MCMC }
9797: procedure WSimFit(RegFunc   : TRegFunc;
9798:                   X, Y, S   : TVector;
9799:                   Lb, Ub    : Integer;
9800:                   B         : TVector;
9801:                   FirstPar,
9802:                   LastPar   : Integer;
9803:                   V         : TMatrix); external 'dmath';
9804: { Simulation of weighted nonlinear regression by MCMC }
9805: { ---------------------------------------------------------------
9806:   Nonlinear regression models
9807:   --------------------------------------------------------------- }
9808: procedure FracFit(X, Y       : TVector;
9809:                   Lb, Ub     : Integer;
9810:                   Deg1, Deg2 : Integer;
9811:                   ConsTerm   : Boolean;
9812:                   MaxIter    : Integer;
9813:                   Tol        : Float;
9814:                   B          : TVector;
9815:                   V          : TMatrix); external 'dmath';
9816: { Unweighted fit of rational fraction }
9817: procedure WFracFit(X, Y, S    : TVector;
9818:                    Lb, Ub     : Integer;
9819:                    Deg1, Deg2 : Integer;
9820:                    ConsTerm   : Boolean;
9821:                    MaxIter    : Integer;
9822:                    Tol        : Float;
9823:                    B          : TVector;
9824:                    V          : TMatrix); external 'dmath';
9825: { Weighted fit of rational fraction }
9826:
9827: function FracFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9828: { Returns the value of the rational fraction at point X }
9829: procedure ExpFit(X, Y         : TVector;
9830:                  Lb, Ub, Nexp : Integer;
9831:                  ConsTerm     : Boolean;
9832:                  MaxIter      : Integer;
9833:                  Tol          : Float;
9834:                  B            : TVector;
9835:                  V            : TMatrix); external 'dmath';
9836: { Unweighted fit of sum of exponentials }
9837: procedure WExpFit(X, Y, S      : TVector;
9838:                   Lb, Ub, Nexp : Integer;
9839:                   ConsTerm     : Boolean;
9840:                   MaxIter      : Integer;
9841:                   Tol          : Float;
9842:                   B            : TVector;
9843:                   V            : TMatrix); external 'dmath';
9844: { Weighted fit of sum of exponentials }
9845: function ExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9846: { Returns the value of the regression function at point X }
9847: procedure IncExpFit(X, Y     : TVector;
9848:                     Lb, Ub   : Integer;
9849:                     ConsTerm : Boolean;
9850:                     MaxIter  : Integer;
9851:                     Tol      : Float;
9852:                     B        : TVector;
9853:                     V        : TMatrix); external 'dmath';
9854: { Unweighted fit of model of increasing exponential }
9855: procedure WIncExpFit(X, Y, S  : TVector;
9856:                      Lb, Ub   : Integer;
9857:                      ConsTerm : Boolean;
9858:                      MaxIter  : Integer;
9859:                      Tol      : Float;
9860:                      B        : TVector;
9861:                      V        : TMatrix); external 'dmath';
9862: { Weighted fit of increasing exponential }
9863: function IncExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9864: { Returns the value of the regression function at point X }
9865: procedure ExpLinFit(X, Y    : TVector;
9866:                     Lb, Ub  : Integer;
9867:                     MaxIter : Integer;
9868:                     Tol     : Float;
9869:                     B       : TVector;
9870:                     V       : TMatrix); external 'dmath';
9871: { Unweighted fit of the "exponential + linear" model }
9872: procedure WExpLinFit(X, Y, S : TVector;
9873:                      Lb, Ub  : Integer;
```

```
                    MaxIter : Integer;
                    Tol     : Float;
                    B       : TVector;
                    V       : TMatrix); external 'dmath';
{ Weighted fit of the "exponential + linear" model }

function ExpLinFit_Func(X : Float; B : TVector) : Float; external 'dmath';
{ Returns the value of the regression function at point X }
procedure MichFit(X, Y    : TVector;
                  Lb, Ub  : Integer;
                  MaxIter : Integer;
                  Tol     : Float;
                  B       : TVector;
                  V       : TMatrix); external 'dmath';
{ Unweighted fit of Michaelis equation }
procedure WMichFit(X, Y, S : TVector;
                   Lb, Ub  : Integer;
                   MaxIter : Integer;
                   Tol     : Float;
                   B       : TVector;
                   V       : TMatrix); external 'dmath';
{ Weighted fit of Michaelis equation }
function MichFit_Func(X : Float; B : TVector) : Float; external 'dmath';
{ Returns the value of the Michaelis equation at point X }
procedure MintFit(X, Y    : TVector;
                  Lb, Ub  : Integer;
                  MintVar : TMintVar;
                  Fit_S0  : Boolean;
                  MaxIter : Integer;
                  Tol     : Float;
                  B       : TVector;
                  V       : TMatrix); external 'dmath';
{ Unweighted fit of the integrated Michaelis equation }
procedure WMintFit(X, Y, S : TVector;
                   Lb, Ub  : Integer;
                   MintVar : TMintVar;
                   Fit_S0  : Boolean;
                   MaxIter : Integer;
                   Tol     : Float;
                   B       : TVector;
                   V       : TMatrix); external 'dmath';
{ Weighted fit of the integrated Michaelis equation }
function MintFit_Func(X : Float; B : TVector) : Float; external 'dmath';
{ Returns the value of the integrated Michaelis equation at point X }
procedure HillFit(X, Y    : TVector;
                  Lb, Ub  : Integer;
                  MaxIter : Integer;
                  Tol     : Float;
                  B       : TVector;
                  V       : TMatrix); external 'dmath';
{ Unweighted fit of Hill equation }
procedure WHillFit(X, Y, S : TVector;
                   Lb, Ub  : Integer;
                   MaxIter : Integer;
                   Tol     : Float;
                   B       : TVector;
                   V       : TMatrix); external 'dmath';
{ Weighted fit of Hill equation }
function HillFit_Func(X : Float; B : TVector) : Float; external 'dmath';
{ Returns the value of the Hill equation at point X }
procedure LogiFit(X, Y     : TVector;
                  Lb, Ub   : Integer;
                  ConsTerm : Boolean;
                  General  : Boolean;
                  MaxIter  : Integer;
                  Tol      : Float;
                  B        : TVector;
                  V        : TMatrix); external 'dmath';
{ Unweighted fit of logistic function }
procedure WLogiFit(X, Y, S  : TVector;
                   Lb, Ub   : Integer;
                   ConsTerm : Boolean;
                   General  : Boolean;
                   MaxIter  : Integer;
                   Tol      : Float;
                   B        : TVector;
                   V        : TMatrix); external 'dmath';
{ Weighted fit of logistic function }
function LogiFit_Func(X : Float; B : TVector) : Float; external 'dmath';
{ Returns the value of the logistic function at point X }
procedure PKFit(X, Y    : TVector;
                Lb, Ub  : Integer;
                MaxIter : Integer;
                Tol     : Float;
                B       : TVector;
                V       : TMatrix); external 'dmath';
{ Unweighted fit of the acid-base titration curve }
procedure WPKFit(X, Y, S : TVector;
                 Lb, Ub  : Integer;
```

```
 9963:                 MaxIter : Integer;
 9964:                 Tol     : Float;
 9965:                 B       : TVector;
 9966:                 V       : TMatrix); external 'dmath';
 9967: { Weighted fit of the acid-base titration curve }
 9968: function PKFit_Func(X : Float; B : TVector) : Float; external 'dmath';
 9969: { Returns the value of the acid-base titration function at point X }
 9970: procedure PowFit(X, Y    : TVector;
 9971:                 Lb, Ub  : Integer;
 9972:                 MaxIter : Integer;
 9973:                 Tol     : Float;
 9974:                 B       : TVector;
 9975:                 V       : TMatrix); external 'dmath';
 9976: { Unweighted fit of power function }
 9977: procedure WPowFit(X, Y, S : TVector;
 9978:                  Lb, Ub  : Integer;
 9979:                  MaxIter : Integer;
 9980:                  Tol     : Float;
 9981:                  B       : TVector;
 9982:                  V       : TMatrix); external 'dmath';
 9983: { Weighted fit of power function }
 9984:
 9985: function PowFit_Func(X : Float; B : TVector) : Float; external 'dmath';
 9986: { Returns the value of the power function at point X }
 9987: procedure GammaFit(X, Y    : TVector;
 9988:                   Lb, Ub  : Integer;
 9989:                   MaxIter : Integer;
 9990:                   Tol     : Float;
 9991:                   B       : TVector;
 9992:                   V       : TMatrix); external 'dmath';
 9993: { Unweighted fit of gamma distribution function }
 9994: procedure WGammaFit(X, Y, S : TVector;
 9995:                    Lb, Ub  : Integer;
 9996:                    MaxIter : Integer;
 9997:                    Tol     : Float;
 9998:                    B       : TVector;
 9999:                    V       : TMatrix); external 'dmath';
10000: { Weighted fit of gamma distribution function }
10001: function GammaFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10002: { Returns the value of the gamma distribution function at point X }
10003: { ---------------------------------------------------------------
10004:   Principal component analysis
10005:   --------------------------------------------------------------- }
10006: procedure VecMean(X            : TMatrix;
10007:                  Lb, Ub, Nvar : Integer;
10008:                  M            : TVector); external 'dmath';
10009: { Computes the mean vector M from matrix X }
10010: procedure VecSD(X            : TMatrix;
10011:                Lb, Ub, Nvar : Integer;
10012:                M, S         : TVector); external 'dmath';
10013: { Computes the vector of standard deviations S from matrix X }
10014: procedure MatVarCov(X             : TMatrix;
10015:                    Lb, Ub, Nvar : Integer;
10016:                    M            : TVector;
10017:                    V            : TMatrix); external 'dmath';
10018: { Computes the variance-covariance matrix V from matrix X }
10019: procedure MatCorrel(V    : TMatrix;
10020:                    Nvar : Integer;
10021:                    R    : TMatrix); external 'dmath';
10022: { Computes the correlation matrix R from the var-cov matrix V }
10023: procedure PCA(R      : TMatrix;
10024:              Nvar   : Integer;
10025:              Lambda : TVector;
10026:              C, Rc  : TMatrix); external 'dmath';
10027: { Performs a principal component analysis of the correlation matrix R }
10028: procedure ScaleVar(X            : TMatrix;
10029:                   Lb, Ub, Nvar : Integer;
10030:                   M, S         : TVector;
10031:                   Z            : TMatrix); external 'dmath';
10032: { Scales a set of variables by subtracting means and dividing by SD's }
10033: procedure PrinFac(Z            : TMatrix;
10034:                  Lb, Ub, Nvar : Integer;
10035:                  C, F         : TMatrix); external 'dmath';
10036: { Computes principal factors }
10037: { ---------------------------------------------------------------
10038:   Strings
10039:   --------------------------------------------------------------- }
10040: function LTrim(S : String) : String; external 'dmath';
10041: { Removes leading blanks }
10042: function RTrim(S : String) : String; external 'dmath';
10043: { Removes trailing blanks }
10044: function Trim(S : String) : String; external 'dmath';
10045: { Removes leading and trailing blanks }
10046: function StrChar(N : Byte; C : Char) : String; external 'dmath';
10047: { Returns a string made of character C repeated N times }
10048: function RFill(S : String; L : Byte) : String; external 'dmath';
10049: { Completes string S with trailing blanks for a total length L }
10050: function LFill(S : String; L : Byte) : String; external 'dmath';
10051: { Completes string S with leading blanks for a total length L }
```

```
10052: function CFill(S : String; L : Byte) : String; external 'dmath';
10053: { Centers string S on a total length L }
10054: function Replace(S : String; C1, C2 : Char) : String; external 'dmath';
10055: { Replaces in string S all the occurences of C1 by C2 }
10056: function Extract(S : String; var Index : Byte; Delim : Char) : String; external 'dmath';
10057: { Extracts a field from a string }
10058: procedure Parse(S : String; Delim:Char; Field:TStrVector; var N:Byte); external 'dmath';
10059: { Parses a string into its constitutive fields }
10060: procedure SetFormat(NumLength,MaxDec:Integer;FloatPoint,NSZero:Bool); external 'dmath';
10061: { Sets the numeric format }
10062: function FloatStr(X : Float) : String; external 'dmath';
10063: { Converts a real to a string according to the numeric format }
10064: function IntStr(N : LongInt) : String; external 'dmath';
10065: { Converts an integer to a string }
10066: function CompStr(Z : Complex) : String; external 'dmath';
10067: { Converts a complex number to a string }
10068: {$IFDEF DELPHI}
10069: function StrDec(S : String) : String; external 'dmath';
10070: { Set decimal separator to the symbol defined in SysUtils }
10071: function IsNumeric(var S : String; var X : Float) : Boolean; external 'dmath';
10072: { Test if a string represents a number and returns it in X }
10073: function ReadNumFromEdit(Edit : TEdit) : Float; external 'dmath';
10074: { Reads a floating point number from an Edit control }
10075: procedure WriteNumToFile(var F : Text; X : Float); external 'dmath';
10076: { Writes a floating point number in a text file }
10077: {$ENDIF}
10078: { ---------------------------------------------------------------------
10079:   BGI / Delphi graphics
10080:   --------------------------------------------------------------------- }
10081: function InitGraphics
10082: {$IFDEF DELPHI}
10083: (Width, Height : Integer) : Boolean;
10084: {$ELSE}
10085: (Pilot, Mode : Integer; BGIPath : String) : Boolean; {$ENDIF} external 'dmath';
10086: { Enters graphic mode }
10087: procedure SetWindow({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10088:                     X1, X2, Y1,Y2 : Integer; GraphBorder:Boolean); external 'dmath';
10089: { Sets the graphic window }
10090: procedure SetOxScale(Scale                : TScale;
10091:                      OxMin, OxMax, OxStep : Float); external 'dmath';
10092: { Sets the scale on the Ox axis }
10093: procedure SetOyScale(Scale                : TScale;
10094:                      OyMin, OyMax, OyStep : Float); external 'dmath';
10095: { Sets the scale on the Oy axis }
10096: procedure GetOxScale(var Scale                : TScale;
10097:                      var OxMin, OxMax, OxStep : Float); external 'dmath';
10098: { Returns the scale on the Ox axis }
10099: procedure GetOyScale(var Scale                : TScale;
10100:                      var OyMin, OyMax, OyStep : Float); external 'dmath';
10101: { Returns the scale on the Oy axis }
10102: procedure SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10103: procedure SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
10104: procedure SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10105: function GetGraphTitle : String; external 'dmath'; { Returns the title for the graph }
10106: function GetOxTitle : String; external 'dmath'; { Returns the title for the Ox axis }
10107: function GetOyTitle : String; external 'dmath'; { Returns the title for the Oy axis }
10108: {$IFNDEF DELPHI}
10109: procedure SetTitleFont(FontIndex, Width, Height : Integer); external 'dmath';
10110: { Sets the font for the main graph title }
10111: procedure SetOxFont(FontIndex, Width, Height : Integer); external 'dmath';
10112: { Sets the font for the Ox axis (title and labels) }
10113: procedure SetOyFont(FontIndex, Width, Height : Integer); external 'dmath';
10114: { Sets the font for the Oy axis (title and labels) }
10115: procedure SetLgdFont(FontIndex, Width, Height : Integer); external 'dmath';
10116: { Sets the font for the legends }
10117: procedure SetClipping(Clip : Boolean); external 'dmath';
10118: { Determines whether drawings are clipped at the current viewport
10119:   boundaries, according to the value of the Boolean parameter Clip }
10120: {$ENDIF}
10121: procedure PlotOxAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10122: { Plots the horizontal axis }
10123: procedure PlotOyAxis{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10124: { Plots the vertical axis }
10125: procedure PlotGrid({$IFDEF DELPHI}Canvas:TCanvas;{$ENDIF} Grid:TGrid); external 'dmath';
10126: { Plots a grid on the graph }
10127: procedure WriteGraphTitle{$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10128: { Writes the title of the graph }
10129: procedure SetMaxCurv(NCurv : Byte); external 'dmath';
10130: { Sets the maximum number of curves and re-initializes their parameters }
10131: procedure SetPointParam
10132: {$IFDEF DELPHI}
10133: (CurvIndex, Symbol, Size : Integer; Color : TColor);
10134: {$ELSE}
10135: (CurvIndex, Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10136: { Sets the point parameters for curve # CurvIndex }
10137: procedure SetLineParam
10138: {$IFDEF DELPHI}
10139: (CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor);
10140: {$ELSE}
```

```
10141: (CurvIndex, Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10142: { Sets the line parameters for curve # CurvIndex }
10143: procedure SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10144: { Sets the legend for curve # CurvIndex }
10145: procedure SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10146: { Sets the step for curve # CurvIndex }
10147: function GetMaxCurv : Byte; external 'dmath'; { Returns the maximum number of curves }
10148: procedure GetPointParam
10149: {$IFDEF DELPHI}
10150: (CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor);
10151: {$ELSE}
10152: (CurvIndex : Integer; var Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10153: { Returns the point parameters for curve # CurvIndex }
10154: procedure GetLineParam
10155: {$IFDEF DELPHI}
10156: (CurvIndex : Integer; var Style : TPenStyle; var Width : Integer; var Color : TColor);
10157: {$ELSE}
10158: (CurvIndex : Integer; var Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10159: { Returns the line parameters for curve # CurvIndex }
10160: function GetCurvLegend(CurvIndex : Integer) : String; external 'dmath';
10161: { Returns the legend for curve # CurvIndex }
10162: function GetCurvStep(CurvIndex : Integer) : Integer; external 'dmath';
10163: { Returns the step for curve # CurvIndex }
10164: {$IFDEF DELPHI}
10165: procedure PlotPoint(Canvas    : TCanvas;
10166:                     X, Y      : Float; CurvIndex : Integer); external 'dmath';
10167: {$ELSE}
10168: procedure PlotPoint(Xp, Yp, CurvIndex : Integer); external 'dmath';
10169: {$ENDIF}
10170: { Plots a point on the screen }
10171: procedure PlotCurve({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10172:                     X, Y                  : TVector;
10173:                    Lb, Ub, CurvIndex     : Integer); external 'dmath';
10174: { Plots a curve }
10175: procedure PlotCurveWithErrorBars({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10176:                                  X, Y, S               : TVector;
10177:                                  Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10178: { Plots a curve with error bars }
10179: procedure PlotFunc({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10180:                    Func                  : TFunc;
10181:                    Xmin, Xmax            : Float;
10182:                    {$IFDEF DELPHI}Npt    : Integer;{$ENDIF}
10183:                    CurvIndex             : Integer); external 'dmath';
10184: { Plots a function }
10185: procedure WriteLegend({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10186:                       NCurv               : Integer;
10187:                       ShowPoints, ShowLines : Boolean); external 'dmath';
10188: { Writes the legends for the plotted curves }
10189: procedure ConRec({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10190:                  Nx, Ny, Nc            : Integer;
10191:                  X, Y, Z               : TVector;
10192:                  F                     : TMatrix); external 'dmath';
10193: { Contour plot }
10194: function Xpixel(X : Float):Integer; external 'dmath'; {Converts user abscissa X to screen coordinate }
10195: function Ypixel(Y : Float):Integer; external 'dmath'; {Converts user ordinate Y to screen coordinate }
10196: function Xuser(X : Integer):Float; external 'dmath';  {Converts screen coordinate X to user abscissa }
10197: function Yuser(Y : Integer):Float; external 'dmath';  {Converts screen coordinate Y to user ordinate }
10198: {$IFNDEF DELPHI}
10199: procedure LeaveGraphics; external 'dmath';
10200: { Quits graphic mode }
10201: {$ENDIF}
10202: { ---------------------------------------------------------------------
10203:   LaTeX graphics
10204:   --------------------------------------------------------------------- }
10205: function TeX_InitGraphics(FileName          : String; PgWidth, PgHeight : Integer;
10206:                           Header            : Boolean) : Boolean; external 'dmath';
10207: { Initializes the LaTeX file }
10208: procedure TeX_SetWindow(X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean); external 'dmath';
10209: { Sets the graphic window }
10210: procedure TeX_LeaveGraphics(Footer : Boolean); external 'dmath'; { Close the LaTeX file }
10211: procedure TeX_SetOxScale(Scale : TScale; OxMin, OxMax, OxStep : Float); external 'dmath';
10212: { Sets the scale on the Ox axis }
10213: procedure TeX_SetOyScale(Scale : TScale; OyMin, OyMax, OyStep : Float); external 'dmath';
10214: { Sets the scale on the Oy axis }
10215: procedure TeX_SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10216: procedure TeX_SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
10217: procedure TeX_SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10218: procedure TeX_PlotOxAxis; external 'dmath'; { Plots the horizontal axis }
10219: procedure TeX_PlotOyAxis; external 'dmath'; { Plots the vertical axis }
10220: procedure TeX_PlotGrid(Grid : TGrid); external 'dmath'; { Plots a grid on the graph }
10221: procedure TeX_WriteGraphTitle; external 'dmath'; Writes the title of the graph }
10222: procedure TeX_SetMaxCurv(NCurv : Byte); external 'dmath';
10223: { Sets the maximum number of curves and re-initializes their parameters }
10224: procedure TeX_SetPointParam(CurvIndex, Symbol, Size : Integer); external 'dmath';
10225: { Sets the point parameters for curve # CurvIndex }
10226: procedure TeX_SetLineParam(CurvIndex, Style : Integer;
10227:                            Width : Float; Smooth : Boolean); external 'dmath';
10228: { Sets the line parameters for curve # CurvIndex }
10229: procedure TeX_SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
```

```
{ Sets the legend for curve # CurvIndex }
procedure TeX_SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
{ Sets the step for curve # CurvIndex }
procedure TeX_PlotCurve(X, Y : TVector; Lb, Ub, CurvIndex : Integer); external 'dmath';
{ Plots a curve }
procedure TeX_PlotCurveWithErrorBars(X, Y, S : TVector;
                                     Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
{ Plots a curve with error bars }
procedure TeX_PlotFunc(Func : TFunc; X1, X2 : Float;
                       Npt : Integer; CurvIndex : Integer); external 'dmath';
{ Plots a function }
procedure TeX_WriteLegend(NCurv : Integer; ShowPoints, ShowLines : Boolean); external 'dmath';
{ Writes the legends for the plotted curves }
procedure TeX_ConRec(Nx, Ny, Nc : Integer; X, Y, Z  : TVector; F : TMatrix); external 'dmath';
{ Contour plot }
function Xcm(X : Float) : Float; external 'dmath'; { Converts user coordinate X to cm }
function Ycm(Y : Float) : Float; external 'dmath'; { Converts user coordinate Y to cm }

//*****************************************************unit uPSI_SynPdf;
 Function RawUTF8ToPDFString( const Value : RawUTF8) : PDFString
 Function _DateTimeToPdfDate( ADate : TDateTime) : TPdfDate
 Function _PdfDateToDateTime( const AText : TPdfDate) : TDateTime
 Function PdfRect( Left, Top, Right, Bottom : Single) : TPdfRect;
 Function PdfRect1( const Box : TPdfBox) : TPdfRect;
 Function PdfBox( Left, Top, Width, Height : Single) : TPdfBox
 //Function _GetCharCount( Text : PAnsiChar) : integer
 //Procedure L2R( W : PWideChar; L : integer)
 Function PdfCoord( MM : single) : integer
 Function CurrentPrinterPaperSize : TPDFPaperSize
 Function CurrentPrinterRes : TPoint
 Procedure GDICommentBookmark( MetaHandle : HDC; const aBookmarkName : RawUTF8)
 Procedure GDICommentOutline( MetaHandle : HDC; const aTitle : RawUTF8; aLevel : Integer)
 Procedure GDICommentLink( MetaHandle:HDC; const aBookmarkName:RawUTF8; const aRect : TRect)
 Const('Usp10','String').SetString( 'usp10.dll
  AddTypeS('TScriptState_enum', '( r0, r1, r2, r3, r4, fOverrideDirection, fInhibitSymSwap,
   'fCharShape, fDigitSubstitute,fInhibitLigate,fDisplayZWG, fArabicNumContext, fGcpClusters )
  AddTypeS('TScriptState_set', 'set of TScriptState_enum
//**********************************************************************

procedure SIRegister_PMrand(CL: TPSPascalCompiler);  //ParkMiller
begin
 Procedure PMrandomize( I : word)
 Function PMrandom : longint
 Function Rrand : extended
 Function Irand( N : word) : word
 Function Brand( P : extended) : boolean
 Function Nrand : extended
end;

procedure SIRegister_Spring_Cryptography_Utils(CL: TPSPascalCompiler);
begin
  Function Endian( x : LongWord) : LongWord
  Function Endian64( x : Int64) : Int64
  Function spRol( x : LongWord; y : Byte) : LongWord
  Function spRor( x : LongWord; y : Byte) : LongWord
  Function Ror64( x : Int64; y : Byte) : Int64
end;

procedure SIRegister_MapReader(CL: TPSPascalCompiler);
begin
 Procedure ClearModules
 Procedure ReadMapFile( Fname : string)
 Function AddressInfo( Address : dword) : string
end;

procedure SIRegister_LibTar(CL: TPSPascalCompiler);
begin
  TTarPermission', '( tpReadByOwner, tpWriteByOwner, tpExecuteByOw'
   +'ner, tpReadByGroup, tpWriteByGroup, tpExecuteByGroup, tpReadByOther, tpWri'
   +'teByOther, tpExecuteByOther )
  TTarPermissions', 'set of TTarPermission
  TFileType', '( ftNormal, ftLink, ftSymbolicLink, ftCharacter, ft'
   +'Block, ftDirectory, ftFifo, ftContiguous, ftDumpDir, ftMultiVolume, ftVolumeHeader;
  TTarMode', '( tmSetUid, tmSetGid, tmSaveText )
  TTarModes', 'set of TTarMode
  TTarDirRec', 'record Name : STRING; Size : INT64; DateTime : TDa'
   +'teTime; Permissions : TTarPermissions; FileType : TFileType; LinkName : ST'
   +'RING; UID : INTEGER; GID : INTEGER; UserName : STRING; GroupName : STRING;'
   +' ChecksumOK : BOOLEAN; Mode : TTarModes; Magic : STRING; MajorDevNo : INTE'
   +'GER; MinorDevNo : INTEGER; FilePos : INT64; end
  SIRegister_TTarArchive(CL);
  SIRegister_TTarWriter(CL);
 Function PermissionString( Permissions : TTarPermissions) : STRING
 Function ConvertFilename( Filename : STRING) : STRING
 Function FileTimeGMT( FileName : STRING) : TDateTime;
 Function FileTimeGMT1( SearchRec : TSearchRec) : TDateTime;
 Procedure ClearDirRec( var DirRec : TTarDirRec)
end;

```

```
10319:
10320: //************************************************unit uPSI_TlHelp32;
10321: procedure SIRegister_TlHelp32(CL: TPSPascalCompiler);
10322: begin
10323:  Const('MAX_MODULE_NAME32','LongInt').SetInt( 255);
10324:  Function CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD) : THandle
10325:  Const('TH32CS_SNAPHEAPLIST','LongWord').SetUInt( $00000001);
10326:  Const('TH32CS_SNAPPROCESS','LongWord').SetUInt( $00000002);
10327:  Const('TH32CS_SNAPTHREAD','LongWord').SetUInt( $00000004);
10328:  Const('TH32CS_SNAPMODULE','LongWord').SetUInt( $00000008);
10329:  Const('TH32CS_INHERIT','LongWord').SetUInt( $80000000);
10330:   tagHEAPLIST32','record dwSize:DWORD;th32ProcessID:DWORD;th32HeapID:DWORD;dwFlags:DWORD;end';
10331:   AddTypeS('HEAPLIST32', 'tagHEAPLIST32
10332:   AddTypeS('THeapList32', 'tagHEAPLIST32
10333:  Const('HF32_DEFAULT','LongInt').SetInt( 1);
10334:  Const('HF32_SHARED','LongInt').SetInt( 2);
10335:  Function Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10336:  Function Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10337:  AddTypeS('tagHEAPENTRY32', 'record dwSize : DWORD; hHandle : THandle; dwAd'
10338:    +'dress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
10339:    +'dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end
10340:   AddTypeS('HEAPENTRY32', 'tagHEAPENTRY32
10341:   AddTypeS('THeapEntry32', 'tagHEAPENTRY32
10342:  Const('LF32_FIXED','LongWord').SetUInt( $00000001);
10343:  Const('LF32_FREE','LongWord').SetUInt( $00000002);
10344:  Const('LF32_MOVEABLE','LongWord').SetUInt( $00000004);
10345:  Function Heap32First( var lphe : THeapEntry32; th32ProcessID, th32HeapID : DWORD) : BOOL
10346:  Function Heap32Next( var lphe : THeapEntry32) : BOOL
10347:  DWORD; var lpNumberOfBytesRead : DWORD) : BOOL
10348:   AddTypeS('tagTHREADENTRY32', 'record dwSize : DWORD; cntUsage : DWORD; th3'
10349:    +'2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt'
10350:    +'aPri : Longint; dwFlags : DWORD; end
10351:   AddTypeS('THREADENTRY32', 'tagTHREADENTRY32
10352:   AddTypeS('TThreadEntry32', 'tagTHREADENTRY32
10353:  Function Thread32First( hSnapshot : THandle; var lpte : TThreadEntry32) : BOOL
10354:  Function Thread32Next( hSnapshot : THandle; var lpte : TThreadENtry32) : BOOL
10355: end;
10356:  Const('EW_RESTARTWINDOWS','LongWord').SetUInt( $0042);
10357:  Const('EW_REBOOTSYSTEM','LongWord').SetUInt( $0043);
10358:  Const('EW_EXITANDEXECAPP','LongWord').SetUInt( $0044);
10359:  Const('ENDSESSION_LOGOFF','LongWord').SetUInt( DWORD ( $80000000 ));
10360:  Const('EWX_LOGOFF','LongInt').SetInt( 0);
10361:  Const('EWX_SHUTDOWN','LongInt').SetInt( 1);
10362:  Const('EWX_REBOOT','LongInt').SetInt( 2);
10363:  Const('EWX_FORCE','LongInt').SetInt( 4);
10364:  Const('EWX_POWEROFF','LongInt').SetInt( 8);
10365:  Const('EWX_FORCEIFHUNG','LongWord').SetUInt( $10);
10366:  Function GET_APPCOMMAND_LPARAM( const lParam : LongInt) : Shortint
10367:  Function GET_DEVICE_LPARAM( const lParam : LongInt) : Word
10368:  Function GET_MOUSEORKEY_LPARAM( const lParam : LongInt) : Word
10369:  Function GET_FLAGS_LPARAM( const lParam : LongInt) : Word
10370:  Function GET_KEYSTATE_LPARAM( const lParam : LongInt) : Word
10371:  Function GetWindowWord( hWnd : HWND; nIndex : Integer) : Word
10372:  Function SetWindowWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10373:  Function GetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
10374:  Function SetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
10375:  Function GetClassWord( hWnd : HWND; nIndex : Integer) : Word
10376:  Function SetClassWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10377:  Function GetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
10378:  Function SetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
10379:  Function GetDesktopWindow : HWND
10380:  Function GetParent( hWnd : HWND) : HWND
10381:  Function SetParent( hWndChild, hWndNewParent : HWND) : HWND
10382:  Function GetTopWindow( hWnd : HWND) : HWND
10383:  Function GetNextWindow( hWnd : HWND; uCmd : UINT) : HWND
10384:  Function GetWindow( hWnd : HWND; uCmd : UINT) : HWND
10385:  //Delphi DFM
10386:  Function LoadDFMFile2Strings(const AFile:string; AStrings:TStrings; var WasText:boolean):integer
10387:  Function SaveStrings2DFMFile( AStrings : TStrings; const AFile : string) : integer
10388:  procedure GetHighlighters(AOwner: TComponent; AHighlighters: TStringList; AppendToList: boolean);
10389:  function GetHighlightersFilter(AHighlighters: TStringList): string;
10390:  function GetHighlighterFromFileExt(AHighlighters: TStringList;Extension: string):TSynCustomHighlighter;
10391:  Function ShowOwnedPopups( hWnd : HWND; fShow : BOOL) : BOOL
10392:  Function OpenIcon( hWnd : HWND) : BOOL
10393:  Function CloseWindow( hWnd : HWND) : BOOL
10394:  Function MoveWindow( hWnd : HWND; X, Y, nWidth, nHeight : Integer; bRepaint : BOOL) : BOOL
10395:  Function SetWindowPos(hWnd: HWND;hWndInsertAfter:HWND; X,Y,cx,cy : Integer; uFlags : UINT) : BOOL
10396:  Function IsWindowVisible( hWnd : HWND) : BOOL
10397:  Function IsIconic( hWnd : HWND) : BOOL
10398:  Function AnyPopup : BOOL
10399:  Function BringWindowToTop( hWnd : HWND) : BOOL
10400:  Function IsZoomed( hWnd : HWND) : BOOL
10401:  Function IsWindow( hWnd : HWND) : BOOL
10402:  Function IsMenu( hMenu : HMENU) : BOOL
10403:  Function IsChild( hWndParent, hWnd : HWND) : BOOL
10404:  Function DestroyWindow( hWnd : HWND) : BOOL
10405:  Function ShowWindow( hWnd : HWND; nCmdShow : Integer) : BOOL
10406:  Function AnimateWindow( hWnd : HWND; dwTime : DWORD; dwFlags : DWORD) : BOOL
10407:  Function ShowWindowAsync( hWnd : HWND; nCmdShow : Integer) : BOOL
```

```
10408:  Function FlashWindow( hWnd : HWND; bInvert : BOOL) : BOOL
10409:  Function IsWindowUnicode( hWnd : HWND) : BOOL
10410:  Function EnableWindow( hWnd : HWND; bEnable : BOOL) : BOOL
10411:  Function IsWindowEnabled( hWnd : HWND) : BOOL
10412:
10413: procedure SIRegister_IDECmdLine(CL: TPSPascalCompiler);
10414: begin
10415:  const('ShowSetupDialogOptLong','String').SetString( '--setup
10416:  PrimaryConfPathOptLong','String').SetString( '--primary-config-path=
10417:  PrimaryConfPathOptShort','String').SetString( '--pcp=
10418:  SecondaryConfPathOptLong','String').SetString( '--secondary-config-path=
10419:  SecondaryConfPathOptShort','String').SetString( '--scp=
10420:  NoSplashScreenOptLong','String').SetString( '--no-splash-screen
10421:  NoSplashScreenOptShort','String').SetString( '--nsc
10422:  StartedByStartLazarusOpt','String').SetString( '--started-by-startlazarus
10423:  SkipLastProjectOpt','String').SetString( '--skip-last-project
10424:  DebugLogOpt','String').SetString( '--debug-log=
10425:  DebugLogOptEnable','String').SetString( '--debug-enable=
10426:  LanguageOpt','String').SetString( '--language=
10427:  LazarusDirOpt','String').SetString( '--lazarusdir=
10428:  Procedure ParseCommandLine(aCmdLineParams:TStrings;out IDEPid:Int;out ShowSplashScreen:boolean);
10429:  Function GetCommandLineParameters( aCmdLineParams : TStrings; isStartLazarus:Boolean) : string
10430:  Function ExtractPrimaryConfigPath( aCmdLineParams : TStrings) : string
10431:  Function IsHelpRequested : Boolean
10432:  Function IsVersionRequested : boolean
10433:  Function GetLanguageSpecified : string
10434:  Function ParamIsOption( ParamIndex : integer; const Option : string) : boolean
10435:  Function ParamIsOptionPlusValue(ParamIndex:integer;const Option:string;out AValue:string):bool;
10436:  Procedure ParseNoGuiCmdLineParams
10437:  Function ExtractCmdLineFilenames : TStrings
10438: end;
10439:
10440:
10441: procedure SIRegister_LazFileUtils(CL: TPSPascalCompiler);
10442: begin
10443:  Function CompareFilenames( const Filename1, Filename2 : string) : integer
10444:  Function CompareFilenamesIgnoreCase( const Filename1, Filename2 : string) : integer
10445:  Function CompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
10446:  Function CompareFileExt1( const Filename, Ext : string) : integer;
10447:  Function CompareFilenameStarts( const Filename1, Filename2 : string) : integer
10448:  Function CompareFilenames(Filename1:PChar;Len1:integer; Filename2:PChar;Len2:integer):integer
10449:  Function CompareFilenamesP( Filename1, Filename2 : PChar; IgnoreCase : boolean) : integer
10450:  Function DirPathExists( DirectoryName : string) : boolean
10451:  Function DirectoryIsWritable( const DirectoryName : string) : boolean
10452:  Function ExtractFileNameOnly( const AFilename : string) : string
10453:  Function FilenameIsAbsolute( const TheFilename : string) : boolean
10454:  Function FilenameIsWinAbsolute( const TheFilename : string) : boolean
10455:  Function FilenameIsUnixAbsolute( const TheFilename : string) : boolean
10456:  Function ForceDirectory( DirectoryName : string) : boolean
10457:  Procedure CheckIfFileIsExecutable( const AFilename : string)
10458:  Procedure CheckIfFileIsSymlink( const AFilename : string)
10459:  Function FileIsText( const AFilename : string) : boolean
10460:  Function FileIsText2( const AFilename : string; out FileReadable : boolean) : boolean
10461:  Function FilenameIsTrimmed( const TheFilename : string) : boolean
10462:  Function FilenameIsTrimmed2( StartPos : PChar; NameLen : integer) : boolean
10463:  Function TrimFilename( const AFilename : string) : string
10464:  Function ResolveDots( const AFilename : string) : string
10465:  Procedure ForcePathDelims( var FileName : string)
10466:  Function GetForcedPathDelims( const FileName : string) : String
10467:  Function CleanAndExpandFilename( const Filename : string) : string
10468:  Function CleanAndExpandDirectory( const Filename : string) : string
10469:  Function TrimAndExpandFilename( const Filename : string; const BaseDir : string) : string
10470:  Function TrimAndExpandDirectory( const Filename : string; const BaseDir : string) : string
10471:  Function TryCreateRelativePath(const Dest,Source:String; UsePointDirectory:bool;
       AlwaysRequireSharedBaseFolder : Boolean; out RelPath : String) : Boolean
10472:  Function CreateRelativePath( const Filename,BaseDirectory:string; UsePointDirectory:boolean;
       AlwaysRequireSharedBaseFolder: Boolean) : string
10473:  Function FileIsInPath( const Filename, Path : string) : boolean
10474:  Function AppendPathDelim( const Path : string) : string
10475:  Function ChompPathDelim( const Path : string) : string
10476:  Function CreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
10477:  Function CreateRelativeSearchPath( const SearchPath, BaseDirectory : string) : string
10478:  Function MinimizeSearchPath( const SearchPath : string) : string
10479:  Function FindPathInSearchPath(APath:PChar;APathLen:int;SearchPath:PChar;SearchPathLen:int):PChar;
10480:  (*Function FileExistsUTF8( const Filename : string) : boolean
10481:  Function FileAgeUTF8( const FileName : string) : Longint
10482:  Function DirectoryExistsUTF8( const Directory : string) : Boolean
10483:  Function ExpandFileNameUTF8( const FileName : string; BaseDir : string) : string
10484:  Function FindFirstUTF8(const Path:string; Attr: Longint; out Rslt : TSearchRec) : Longint
10485:  Function FindNextUTF8( var Rslt : TSearchRec) : Longint
10486:  Procedure FindCloseUTF8( var F : TSearchrec)
10487:  Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
10488:  Function FileGetAttrUTF8( const FileName : String) : Longint
10489:  Function FileSetAttrUTF8( const Filename : String; Attr : longint) : Longint
10490:  Function DeleteFileUTF8( const FileName : String) : Boolean
10491:  Function RenameFileUTF8( const OldName, NewName : String) : Boolean
10492:  Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
10493:  Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
10494:  Function GetCurrentDirUTF8 : String
```

```
10495:  Function SetCurrentDirUTF8( const NewDir : String) : Boolean
10496:  Function CreateDirUTF8( const NewDir : String) : Boolean
10497:  Function RemoveDirUTF8( const Dir : String) : Boolean
10498:  Function ForceDirectoriesUTF8( const Dir : string) : Boolean
10499:  Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
10500:  Function FileCreateUTF8( const FileName : string) : THandle;
10501:  Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
10502:  Function FileCreateUtf82( const FileName : String; ShareMode : Integer; Rights : Cardinal) : THandle;
10503:  Function FileSizeUtf8( const Filename : string) : int64
10504:  Function GetFileDescription( const AFilename : string) : string
10505:  Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
10506:  Function GetAppConfigFileUTF8( Global : Boolean; SubDir : boolean; CreateDir : boolean) : string
10507:  Function GetTempFileNameUTF8( const Dir, Prefix : String) : String*)
10508:  Function IsUNCPath( const Path : String) : Boolean
10509:  Function ExtractUNCVolume( const Path : String) : String
10510:  Function ExtractFileRoot( FileName : String) : String
10511:  Function GetDarwinSystemFilename( Filename : string) : string
10512:  Procedure SplitCmdLineParams( const Params : string; ParamList : TStrings; ReadBackslash : boolean)
10513:  Function StrToCmdLineParam( const Param : string) : string
10514:  Function MergeCmdLineParams( ParamList : TStrings) : string
10515:  Procedure InvalidateFileStateCache( const Filename : string)
10516:  Function FindAllFiles(const SearchPath:String;SearchMask:String;SearchSubDirs:Boolean):TStringList);
10517:  Function FindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
10518:  Function ReadFileToString( const Filename : string) : string
10519:  type
10520:   TCopyFileFlag = ( cffOverwriteFile,
10521:     cffCreateDestDirectory, cffPreserveTime );
10522:   TCopyFileFlags = set of TCopyFileFlag;*)
10523:   TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime)
10524:   TCopyFileFlags', 'set of TCopyFileFlag
10525:   Function CopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
10526: end;
10527:
10528: procedure SIRegister_lazMasks(CL: TPSPascalCompiler);
10529: begin
10530:   TMaskCharType', '( mcChar, mcCharSet, mcAnyChar, mcAnyText )
10531:   SIRegister_TMask(CL);
10532:   SIRegister_TParseStringList(CL);
10533:   SIRegister_TMaskList(CL);
10534:  Function MatchesMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Boolean
10535:  Function MatchesWindowsMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Bool;
10536:  Function MatchesMaskList(const FileName,Mask:String;Separator:Char;const CaseSensitive:Boolean):Bool;
10537:  Function MatchesWindowsMaskList(const FileName,Mask:String;Separat:Char;const CaseSensitive:Bool):Bool;
10538: end;
10539:
10540: procedure SIRegister_JvShellHook(CL: TPSPascalCompiler);
10541: begin
10542:   //PShellHookInfo', '^TShellHookInfo // will not work
10543:   TShellHookInfo', 'record hwnd : THandle; rc : TRect; end
10544:   SHELLHOOKINFO', 'TShellHookInfo
10545:   LPSHELLHOOKINFO', 'PShellHookInfo
10546:   TJvShellHookEvent', 'Procedure ( Sender : TObject; var Msg : TMessage)
10547:   SIRegister_TJvShellHook(CL);
10548:  Function InitJvShellHooks : Boolean
10549:  Procedure UnInitJvShellHooks
10550: end;
10551:
10552: procedure SIRegister_JvExControls(CL: TPSPascalCompiler);
10553: begin
10554:   TDlgCode', '( dcWantAllKeys, dcWantArrows, dcWantChars, dcButton'
10555:    +', dcHasSetSel, dcWantTab, dcNative )
10556:   TDlgCodes', 'set of TDlgCode
10557:  'dcWantMessage','').SetString( dcWantAllKeys);
10558:   SIRegister_IJvExControl(CL);
10559:   SIRegister_IJvDenySubClassing(CL);
10560:   SIRegister_TStructPtrMessage(CL);
10561:  Procedure SetDotNetFrameColors( FocusedColor, UnfocusedColor : TColor)
10562:  Procedure DrawDotNetControl( Control : TWinControl; AColor : TColor; InControl : Boolean);
10563:  Procedure DrawDotNetControl1( DC : HDC; R : TRect; AColor : TColor; UseFocusedColor : Boolean);
10564:  Procedure HandleDotNetHighlighting(Control:TWinControl;const Msg:TMessage;MouseOver:Boolean;Color:TColor);
10565:  Function CreateWMMessage( Msg : Integer; WParam : Integer; LParam : Longint) : TMessage;
10566:  Function CreateWMMessage1( Msg : Integer; WParam : Integer; LParam : TControl) : TMessage;
10567:  Function SmallPointToLong( const Pt : TSmallPoint) : Longint
10568:  Function ShiftStateToKeyData( Shift : TShiftState) : Longint
10569:  Function GetFocusedControl( AControl : TControl) : TWinControl
10570:  Function DlgcToDlgCodes( Value : Longint) : TDlgCodes
10571:  Function DlgCodesToDlgc( Value : TDlgCodes) : Longint
10572:  Procedure GetHintColor( var HintInfo : THintInfo; AControl : TControl; HintColor : TColor)
10573:  Function DispatchIsDesignMsg( Control : TControl; var Msg : TMessage) : Boolean
10574:   SIRegister_TJvExControl(CL);
10575:   SIRegister_TJvExWinControl(CL);
10576:   SIRegister_TJvExCustomControl(CL);
10577:   SIRegister_TJvExGraphicControl(CL);
10578:   SIRegister_TJvExHintWindow(CL);
10579:   SIRegister_TJvExPubGraphicControl(CL);
10580: end;
10581:
10582: (*----------------------------------------------------------------------------*)
10583: procedure SIRegister_EncdDecd(CL: TPSPascalCompiler);
```

```
10584: begin
10585:  Procedure EncodeStream( Input, Output : TStream)
10586:  Procedure DecodeStream( Input, Output : TStream)
10587:  Function EncodeString1( const Input : string) : string
10588:  Function DecodeString1( const Input : string) : string
10589: end;
10590:
10591: (*----------------------------------------------------------------------------*)
10592: procedure SIRegister_SockAppReg(CL: TPSPascalCompiler);
10593: begin
10594:   SIRegister_TWebAppRegInfo(CL);
10595:   SIRegister_TWebAppRegList(CL);
10596:  Procedure GetRegisteredWebApps( AList : TWebAppRegList)
10597:  Procedure RegisterWebApp( const AFileName, AProgID : string)
10598:  Procedure UnregisterWebApp( const AProgID : string)
10599:  Function FindRegisteredWebApp( const AProgID : string) : string
10600:  Function CreateRegistry( InitializeNewFile : Boolean) : TCustomIniFile
10601:  'sUDPPort','String').SetString( 'UDPPort
10602: end;
10603:
10604: procedure SIRegister_PJEnvVars(CL: TPSPascalCompiler);
10605: begin
10606:  // TStringDynArray', 'array of string
10607:  Function GetEnvVarValue( const VarName : string) : string
10608:  Function SetEnvVarValue( const VarName, VarValue : string) : Integer
10609:  Function DeleteEnvVar( const VarName : string) : Integer
10610:  Function CreateEnvBlock(const NewEnv:TStrings;const IncludeCurrent:Bool;const Buffer:string;const
      BufSize:Int):Int;
10611:  Function ExpandEnvVars( const Str : string) : string
10612:  Function GetAllEnvVars( const Vars : TStrings) : Integer
10613:  Procedure GetAllEnvVarNames( const Names : TStrings);
10614:  Function GetAllEnvVarNames1 : TStringDynArray;
10615:  Function EnvBlockSize : Integer
10616:   TPJEnvVarsEnum', 'Procedure ( const VarName : string; Data : TObject)
10617:   SIRegister_TPJEnvVarsEnumerator(CL);
10618:   SIRegister_TPJEnvVars(CL);
10619:   FindClass('TOBJECT'),'EPJEnvVars
10620:   FindClass('TOBJECT'),'EPJEnvVars
10621:  //Procedure Register
10622: end;
10623:
10624: (*----------------------------------------------------------------------------*)
10625: procedure SIRegister_PJConsoleApp(CL: TPSPascalCompiler);
10626: begin
10627:  'cOneSecInMS','LongInt').SetInt( 1000);
10628:  //'cDefTimeSlice','LongInt').SetInt( 50);
10629:  //'cDefMaxExecTime','').SetString( cOneMinInMS);
10630:  'cAppErrorMask','LongInt').SetInt( 1 shl 29);
10631:  Function IsApplicationError( const ErrCode : LongWord) : Boolean
10632:   TPJConsoleAppPriority', '( cpDefault, cpHigh, cpNormal, cpIdle, cpRealTime )
10633:   TPJConsoleColors', 'record Foreground : TPJConsoleColor; Background:TPJConsoleColor; end;
10634:  Function MakeConsoleColors( const AForeground, ABackground : TPJConsoleColor):TPJConsoleColors;
10635:  Function MakeConsoleColors1( const AForeground, ABackground : TColor) : TPJConsoleColors;
10636:  Function MakeConsoleColors2( const AForeground, ABackground : TAlphaColor) : TPJConsoleColors;
10637:  Function MakeSize( const ACX, ACY : LongInt) : TSize
10638:   SIRegister_TPJCustomConsoleApp(CL);
10639:   SIRegister_TPJConsoleApp(CL);
10640: end;
10641:
10642: procedure SIRegister_ip_misc(CL: TPSPascalCompiler);
10643: begin
10644:  INVALID_IP_ADDRESS','LongWord').SetUInt( $ffffffff);
10645:   t_encoding', '( uuencode, base64, mime )
10646:  Function internet_date( date : TDateTime) : string
10647:  Function lookup_hostname( const hostname : string) : longint
10648:  Function my_hostname : string
10649:  Function my_ip_address : longint
10650:  Function ip2string( ip_address : longint) : string
10651:  Function resolve_hostname( ip : longint) : string
10652:  Function address_from( const s : string; count : integer) : string
10653:  Function encode_base64( data : TStream) : TStringList
10654:  Function decode_base64( source : TStringList) : TMemoryStream
10655:  Function posn( const s, t : string; count : integer) : integer
10656:  Function poscn( c : char; const s : string; n : integer) : integer
10657:  Function filename_of( const s : string) : string
10658:  //Function trim( const s : string) : string
10659:  //Procedure setlength( var s : string; l : byte)
10660:  Function TimeZoneBias : longint
10661:  Function eight2seven_quoteprint( const s : string) : string
10662:  Function eight2seven_german( const s : string) : string
10663:  Function seven2eight_quoteprint( const s : string) : string end;
10664:   type in_addr', 'record s_bytes : array[1..4] of byte; end;
10665:  Function socketerror : cint
10666:  Function fpsocket( domain : cint; xtype : cint; protocol : cint) : cint
10667:  Function fprecv( s : cint; buf : ___pointer; len : size_t; flags : cint) : ssize_t
10668:  Function fpsend( s : cint; msg : ___pointer; len : size_t; flags : cint) : ssize_t
10669:  //Function fpbind( s : cint; addrx : psockaddr; addrlen : tsocklen) : cint
10670:  Function fplisten( s : cint; backlog : cint) : cint
10671:  //Function fpaccept( s : cint; addrx : psockaddr; addrlen : plongint) : cint
```

```
10672:  //Function fpconnect( s : cint; name : psockaddr; namelen : tsocklen) : cint
10673:  //Function fpgetsockname( s : cint; name : psockaddr; namelen : psocklen) : cint
10674:  Function NetAddrToStr( Entry : in_addr) : String
10675:  Function HostAddrToStr( Entry : in_addr) : String
10676:  Function StrToHostAddr( IP : String) : in_addr
10677:  Function StrToNetAddr( IP : String) : in_addr
10678:  SOL_SOCKET','LongWord').SetUInt( $ffff);
10679:   cint8', 'shortint
10680:   cuint8', 'byte
10681:   cchar', 'cint8
10682:   cschar', 'cint8
10683:   cuchar', 'cuint8
10684:   cint16', 'smallint
10685:   cuint16', 'word
10686:   cshort', 'cint16
10687:   csshort', 'cint16
10688:   cushort', 'cuint16
10689:   cint32', 'longint
10690:   cuint32', 'longword
10691:   cint', 'cint32
10692:   csint', 'cint32
10693:   cuint', 'cuint32
10694:   csigned', 'cint
10695:   cunsigned', 'cuint
10696:   cint64', 'int64
10697:   clonglong', 'cint64
10698:   cslonglong', 'cint64
10699:   cbool', 'longbool
10700:   cfloat', 'single
10701:   cdouble', 'double
10702:   clongdouble', 'extended
10703:
10704: procedure SIRegister_uLkJSON(CL: TPSPascalCompiler);
10705: begin
10706:   TlkJSONtypes','(jsBase,jsNumber,jsString,jsBoolean,jsNull,jsList,jsObject )
10707:   SIRegister_TlkJSONdotnetclass(CL);
10708:   SIRegister_TlkJSONbase(CL);
10709:   SIRegister_TlkJSONnumber(CL);
10710:   SIRegister_TlkJSONstring(CL);
10711:   SIRegister_TlkJSONboolean(CL);
10712:   SIRegister_TlkJSONnull(CL);
10713:   TlkJSONFuncEnum', 'Procedure ( ElName : string; Elem : TlkJSONba'
10714:    +'se; data : TObject; var Continue : Boolean)
10715:   SIRegister_TlkJSONcustomlist(CL);
10716:   SIRegister_TlkJSONlist(CL);
10717:   SIRegister_TlkJSONobjectmethod(CL);
10718:   TlkHashItem', 'record hash : cardinal; index : Integer; end
10719:   TlkHashFunction', 'Function ( const ws : WideString) : cardinal
10720:   SIRegister_TlkHashTable(CL);
10721:   SIRegister_TlkBalTree(CL);
10722:   SIRegister_TlkJSONobject(CL);
10723:   SIRegister_TlkJSON(CL);
10724:   SIRegister_TlkJSONstreamed(CL);
10725:  Function GenerateReadableText( vObj : TlkJSONbase; var vLevel : Integer): string
10726: end;
10727:
10728: procedure SIRegister_ZSysUtils(CL: TPSPascalCompiler);
10729: begin
10730:   TZListSortCompare', 'Function (Item1, Item2 : TObject): Integer
10731:   SIRegister_TZSortedList(CL);
10732:  Function zFirstDelimiter( const Delimiters, Str : string) : Integer
10733:  Function zLastDelimiter( const Delimiters, Str : string) : Integer
10734:  //Function MemLCompUnicode( P1, P2 : PWideChar; Len : Integer) : Boolean
10735:  //Function MemLCompAnsi( P1, P2 : PAnsiChar; Len : Integer) : Boolean
10736:  Function zStartsWith( const Str, SubStr : WideString) : Boolean;
10737:  Function StartsWith1( const Str, SubStr : RawByteString) : Boolean;
10738:  Function EndsWith( const Str, SubStr : WideString) : Boolean;
10739:  Function EndsWith1( const Str, SubStr : RawByteString) : Boolean;
10740:  Function SQLStrToFloatDef( Str : RawByteString; Def : Extended) : Extended;
10741:  Function SQLStrToFloatDef1( Str : String; Def : Extended) : Extended;
10742:  Function SQLStrToFloat( const Str : AnsiString) : Extended
10743:  //Function BufferToStr( Buffer : PWideChar; Length : LongInt) : string;
10744:  //Function BufferToStr1( Buffer : PAnsiChar; Length : LongInt) : string;
10745:  Function BufferToBytes( Buffer : TObject; Length : LongInt) : TByteDynArray
10746:  Function StrToBoolEx( Str : string) : Boolean
10747:  Function BoolToStrEx( Bool : Boolean) : String
10748:  Function IsIpAddr( const Str : string) : Boolean
10749:  Function zSplitString( const Str, Delimiters : string) : TStrings
10750:  Procedure PutSplitString( List : TStrings; const Str, Delimiters : string)
10751:  Procedure AppendSplitString( List : TStrings; const Str, Delimiters : string)
10752:  Function ComposeString( List : TStrings; const Delimiter : string) : string
10753:  Function FloatToSQLStr( Value : Extended) : string
10754:  Procedure PutSplitStringEx( List : TStrings; const Str, Delimiter : string)
10755:  Function SplitStringEx( const Str, Delimiter : string) : TStrings
10756:  Procedure AppendSplitStringEx( List : TStrings; const Str, Delimiter : string)
10757:  Function zBytesToStr( const Value : TByteDynArray) : AnsiString
10758:  Function zStrToBytes( const Value : AnsiString) : TByteDynArray;
10759:  Function StrToBytes1( const Value : UTF8String) : TByteDynArray;
10760:  Function StrToBytes2( const Value : RawByteString) : TByteDynArray;
```

```
10761:  Function StrToBytes3( const Value : WideString) : TByteDynArray;
10762:  Function StrToBytes4( const Value : UnicodeString) : TByteDynArray;
10763:  Function BytesToVar( const Value : TByteDynArray) : Variant
10764:  Function VarToBytes( const Value : Variant) : TByteDynArray
10765:  Function AnsiSQLDateToDateTime( const Value : string) : TDateTime
10766:  Function TimestampStrToDateTime( const Value : string) : TDateTime
10767:  Function DateTimeToAnsiSQLDate( Value : TDateTime; WithMMSec : Boolean) : string
10768:  Function EncodeCString( const Value : string) : string
10769:  Function DecodeCString( const Value : string) : string
10770:  Function zReplaceChar( const Source, Target : Char; const Str : string) : string
10771:  Function MemPas( Buffer : PChar; Length : LongInt) : string
10772:  Procedure DecodeSQLVersioning(const FullVersion:Int;out MajorVersion:Int;out MinorVersion:Int;out
       SubVersion:Int);
10773:  Function EncodeSQLVersioning(const MajorVersion:Integer;const MinorVersion:Integer;const
       SubVersion:Integer):Int;
10774:  Function FormatSQLVersion( const SQLVersion : Integer) : String
10775:  Function ZStrToFloat( Value : AnsiChar) : Extended;
10776:  Function ZStrToFloat1( Value : AnsiString) : Extended;
10777:  Procedure ZSetString( const Src : AnsiChar; var Dest : AnsiString);
10778:  Procedure ZSetString1( const Src : AnsiChar; const Len : Cardinal; var Dest : AnsiString);
10779:  Procedure ZSetString2( const Src : AnsiChar; var Dest : UTF8String);
10780:  Procedure ZSetString3( const Src : AnsiChar; const Len : Cardinal; var Dest : UTF8String);
10781:  Procedure ZSetString4( const Src : AnsiChar; const Len : Cardinal; var Dest : WideString);
10782:  Procedure ZSetString5( const Src : AnsiChar; var Dest : RawByteString);
10783:  Procedure ZSetString6( const Src : AnsiChar; const Len : Cardinal; var Dest : RawByteString);
10784: end;
10785:
10786: unit uPSI_ZEncoding;
10787:  Function StringToAnsiEx( const s : String; const FromCP, ToCP : Word) : RawByteString
10788:  Function AnsiToStringEx( const s : RawByteString; const FromCP, ToCP : Word) : String
10789:  Function ZRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10790:  Function ZUnicodeToRaw( const US : WideString; CP : Word) : RawByteString
10791:  Function ZConvertAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10792:  Function ZConvertRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10793:  Function ZConvertAnsiToUTF8( const Src : AnsiString) : UTF8String
10794:  Function ZConvertUTF8ToAnsi( const Src : UTF8String) : AnsiString
10795:  Function ZConvertRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10796:  Function ZConvertUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10797:  Function ZConvertRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10798:  Function ZConvertStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10799:  Function ZConvertStringToRawWithAutoEncode(const Src:String;const StringCP,RawCP:Word):RawByteString;
10800:  Function ZConvertUTF8ToString( const Src : UTF8String; const StringCP : Word) : String
10801:  Function ZConvertStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10802:  Function ZConvertStringToUTF8WithAutoEncode( const Src : String; const StringCP: Word): UTF8String
10803:  Function ZConvertStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10804:  Function ZConvertStringToAnsiWithAutoEncode( const Src : String; const StringCP: Word): AnsiString
10805:  Function ZConvertAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10806:  Function ZConvertUnicodeToString( const Src : WideString; const StringCP : Word) : String
10807:  Function ZConvertUnicodeToString_CPUTF8( const Src : WideString; const StringCP : Word) : String
10808:  Function ZConvertStringToUnicode( const Src : String; const StringCP : Word) : WideString
10809:  Function ZConvertString_CPUTF8ToUnicode( const Src : String; const StringCP : Word) : WideString
10810:  Function ZConvertStringToUnicodeWithAutoEncode( const Src: String; const StringCP:Word):WideString
10811:  Function ZMoveAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10812:  Function ZMoveRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10813:  Function ZMoveAnsiToUTF8( const Src : AnsiString) : UTF8String
10814:  Function ZMoveUTF8ToAnsi( const Src : UTF8String) : AnsiString
10815:  Function ZMoveRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10816:  Function ZMoveUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10817:  Function ZMoveStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10818:  Function ZMoveAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10819:  Function ZMoveRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10820:  Function ZMoveStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10821:  Function ZMoveUTF8ToString( const Src : UTF8String; StringCP : Word) : String
10822:  Function ZMoveStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10823:  Function ZUnknownRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10824:  Function ZUnknownRawToUnicodeWithAutoEncode( const S : RawByteString; const CP : Word) : WideString
10825:  Function ZUnicodeToUnknownRaw( const US : WideString; CP : Word) : RawByteString
10826:  Function ZDefaultSystemCodePage : Word
10827:  Function ZCompatibleCodePages( const CP1, CP2 : Word) : Boolean
10828:
10829:
10830: procedure SIRegister_BoldComUtils(CL: TPSPascalCompiler);
10831: begin
10832:  'RPC_C_AUTHN_LEVEL_DEFAULT','LongInt').SetInt( 0);
10833:  ('RPC_C_AUTHN_LEVEL_NONE','LongInt').SetInt( 1);
10834:  ('RPC_C_AUTHN_LEVEL_CONNECT','LongInt').SetInt( 2);
10835:  ('RPC_C_AUTHN_LEVEL_CALL','LongInt').SetInt( 3);
10836:  ('RPC_C_AUTHN_LEVEL_PKT','LongInt').SetInt( 4);
10837:  ('RPC_C_AUTHN_LEVEL_PKT_INTEGRITY','LongInt').SetInt( 5);
10838:  ('RPC_C_AUTHN_LEVEL_PKT_PRIVACY','LongInt').SetInt( 6);
10839:  {('alDefault','1').SetString( RPC_C_AUTHN_LEVEL_DEFAULT);
10840:  ('alNone','2').SetString( RPC_C_AUTHN_LEVEL_NONE);
10841:  ('alConnect','3').SetString( RPC_C_AUTHN_LEVEL_CONNECT);
10842:  ('alCall','4').SetString( RPC_C_AUTHN_LEVEL_CALL);
10843:  ('alPacket','5').SetString( RPC_C_AUTHN_LEVEL_PKT);
10844:  ('alPacketIntegrity','6').SetString( RPC_C_AUTHN_LEVEL_PKT_INTEGRITY);
10845:  ('alPacketPrivacy','7').SetString( RPC_C_AUTHN_LEVEL_PKT_PRIVACY);}
10846:  ('RPC_C_IMP_LEVEL_DEFAULT','LongInt').SetInt( 0);
10847:  ('RPC_C_IMP_LEVEL_ANONYMOUS','LongInt').SetInt( 1);
```

```
10848:  ('RPC_C_IMP_LEVEL_IDENTIFY','LongInt').SetInt( 2);
10849:  ('RPC_C_IMP_LEVEL_IMPERSONATE','LongInt').SetInt( 3);
10850:  ('RPC_C_IMP_LEVEL_DELEGATE','LongInt').SetInt( 4);
10851:  {('ilDefault','0').SetString( RPC_C_IMP_LEVEL_DEFAULT);
10852:  ('ilAnonymous','1').SetString( RPC_C_IMP_LEVEL_ANONYMOUS);
10853:  ('ilIdentiry','2').SetString( RPC_C_IMP_LEVEL_IDENTIFY);
10854:  ('ilImpersonate','3').SetString( RPC_C_IMP_LEVEL_IMPERSONATE);
10855:  ('ilDelegate','4').SetString( RPC_C_IMP_LEVEL_DELEGATE);}
10856:  ('EOAC_NONE','LongWord').SetUInt( $0);
10857:  ('EOAC_DEFAULT','LongWord').SetUInt( $800);
10858:  ('EOAC_MUTUAL_AUTH','LongWord').SetUInt( $1);
10859:  ('EOAC_STATIC_CLOACKING','LongWord').SetUInt( $20);
10860:  ('EOAC_DYNAMIC_CLOAKING','LongWord').SetUInt( $40);
10861:  ('EOAC_ANY_AUTHORITY','LongWord').SetUInt( $80);
10862:  ('RPC_C_AUTHN_WINNT','LongInt').SetInt( 10);
10863:  ('RPC_C_AUTHNZ_NONE','LongInt').SetInt( 0);
10864:  ('RPC_C_AUTHNZ_NAME','LongInt').SetInt( 1);
10865:  ('RPC_C_AUTHNZ_DCE','LongInt').SetInt( 2);
10866:   FindClass('TOBJECT'),'EBoldCom
10867:  Function BoldVariantIsType( V : OleVariant; TypeCode : Integer) : Boolean
10868:  Function BoldMemoryToVariant( const Buffer, BufSize : Integer) : OleVariant
10869:  Function BoldStreamToVariant( Stream : TStream) : OleVariant
10870:  Function BoldStringsToVariant( Strings : TStrings) : OleVariant
10871:  Function BoldVariantToMemory( V : OleVariant; var Buffer, BufSize : Integer) : Integer
10872:  Function BoldVariantToStream( V : OleVariant; Stream : TStream) : Integer
10873:  Function BoldVariantArrayOfArraysOfStringToStrings( V : OleVariant; Strings : TStrings) : Integer
10874:  Function BoldVariantIsNamedValues( V : OleVariant) : Boolean
10875:  Function BoldCreateNamedValues(const Names:array of string;const Values:array of OleVariant):OleVariant;
10876:  Function BoldGetNamedValue( Data : OleVariant; const Name : string) : OleVariant
10877:  Procedure BoldSetNamedValue( Data : OleVariant; const Name : string; Value : OleVariant)
10878:  Function BoldCreateGUID : TGUID
10879:  Function BoldCreateComObject( const ClsId, IId : TGUID; out Obj : variant; out Res : HResult) : Boolean
10880:  Function BoldCreateRemoteComObject(const HostName:string;const ClsId,IId:TGUID;out Obj:variant;out
       Res:HRes):Bool;
10881:  Procedure BoldInitializeComSecurity( AuthenticationLevel, ImpersonationLevel : longint)
10882:  Procedure BoldSetSecurityForInterface(AuthenticationLevel,ImpersonationLevel:longint;Unk:IUnknown);
10883: end;
10884:
10885: (*----------------------------------------------------------------------------*)
10886: procedure SIRegister_BoldIsoDateTime(CL: TPSPascalCompiler);
10887: begin
10888:  Function ParseISODate( s : string) : TDateTime
10889:  Function ParseISODateTime( s : string) : TDateTime
10890:  Function ParseISOTime( str : string) : TDateTime
10891: end;
10892:
10893: (*----------------------------------------------------------------------------*)
10894: procedure SIRegister_BoldGUIDUtils(CL: TPSPascalCompiler);
10895: begin
10896:  Function BoldCreateGUIDAsString( StripBrackets : Boolean) : string
10897:  Function BoldCreateGUIDWithBracketsAsString : string
10898: end;
10899:
10900: procedure SIRegister_BoldFileHandler(CL: TPSPascalCompiler);
10901: begin
10902:   FindClass('TOBJECT'),'TBoldFileHandler
10903:   FindClass('TOBJECT'),'TBoldDiskFileHandler
10904:   //TBoldFileHandlerClass', 'class of TBoldFileHandler
10905:   TBoldInitializeFileContents', 'Procedure ( StringList : TStringList)
10906:   SIRegister_TBoldFileHandler(CL);
10907:   SIRegister_TBoldDiskFileHandler(CL);
10908:  Procedure BoldCloseAllFilehandlers
10909:  Procedure BoldRemoveUnchangedFilesFromEditor
10910:  Function BoldFileHandlerList : TBoldObjectArray
10911:  Function BoldFileHandlerForFile(path,FileName:String; ModuleType:TBoldModuleType;ShowInEditor:Bool;
       OnInitializeFileContents : TBoldInitializeFileContents) : TBoldFileHandler
10912: end;
10913:
10914: procedure SIRegister_BoldWinINet(CL: TPSPascalCompiler);
10915: begin
10916:   PCharArr', 'array of PChar
10917:  Function BoldInternetOpen(Agent:String;
       AccessType:integer;Proxy:string;ProxyByPass:String;Flags:integer):ptr);
10918:  Function BoldInternetOpenUrl(iNet:Pointer;URL: string; Headers:String;Flags,Context:cardinal):Pointer
10919:  Function BoldInternetReadFile(hFile:Pointer;Buff:Ptr;NumberOfBytesToRead:Card;var
       NumberOfBytesRead:Card):LongBool;
10920:  Function BoldInternetCloseHandle( HINet : Pointer) : LongBool
10921:  Function BoldHttpQueryInfo( hRequest : Pointer; InfoLevel : Cardinal; Buffer : Pointer; BufferLength :
       Cardinal; Reserved : Cardinal) : LongBool
10922:  Function BoldInternetQueryDataAvailable( hFile : Pointer; var NumberOfBytesAvailable : Cardinal; flags :
       Cardinal; Context : Cardinal) : LongBool
10923:  Function BoldHttpOpenRequest(hConnect: Pointer; Verb, ObjectName, Version, Referrer : String; AcceptTypes
       : PCharArr; Flags, Context : Cardinal) : Pointer
10924:  Function BoldHttpSendRequest(hRequest:Pointer;Headers:string; Optional:Pointer;OptionalLength:Cardinal):
       LongBool
10925:  Function BoldInternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD; var lppvData:Pointer):
       DWORD
10926:  Function BoldInternetAttemptConnect( dwReserved : DWORD) : DWORD
10927:  Function BoldInternetConnect( hInet : HINTERNET; ServerName : string; nServerPort : INTERNET_PORT;
       Username : string; Password : string; dwService : DWORD; dwFlags : DWORD; dwContext : DWORD) : HINTERNET
```

```
10928:  Function BoldInternetCrackUrl(Url:PChar;UrlLength,dwFlags:DWORD; var lpUrlComponents:TURLComponents):BOOL;
10929: end;
10930:
10931: procedure SIRegister_BoldQueryUserDlg(CL: TPSPascalCompiler);
10932: begin
10933:   TBoldQueryResult', '( qrYesAll, qrYes, qrNo, qrNoAll )
10934:   SIRegister_TfrmBoldQueryUser(CL);
10935:  Function QueryUser( const Title, Query : string) : TBoldQueryResult
10936: end;
10937:
10938: (*----------------------------------------------------------------------------*)
10939: procedure SIRegister_BoldQueue(CL: TPSPascalCompiler);
10940: begin
10941:  //('befIsInDisplayList','').SetString( BoldElementFlag0);
10942:  //('befStronglyDependedOfPrioritized','').SetString( BoldElementFlag1);
10943:  //('befFollowerSelected','').SetString( BoldElementFlag2);
10944:   FindClass('TOBJECT'),'TBoldQueue
10945:   FindClass('TOBJECT'),'TBoldQueueable
10946:   TBoldQueueDisplayMode', '( dmDisplayOne, dmDisplayAll )
10947:   SIRegister_TBoldQueueable(CL);
10948:   SIRegister_TBoldQueue(CL);
10949:  Function BoldQueueFinalized : Boolean
10950:  Function BoldInstalledQueue : TBoldQueue
10951: end;
10952:
10953: procedure SIRegister_Barcode(CL: TPSPascalCompiler);
10954: begin
10955:  const mmPerInch','Extended').setExtended( 25.4);
10956:   TBarcodeType', '( bcCode_2_5_interleaved, bcCode_2_5_industrial,'
10957:    +' bcCode_2_5_matrix, bcCode39, bcCode39Extended, bcCode128A, bcCode128B, bc'
10958:    +'Code128C, bcCode93, bcCode93Extended, bcCodeMSI, bcCodePostNet, bcCodeCoda'
10959:    +'bar, bcCodeEAN8, bcCodeEAN13, bcCodeUPC_A, bcCodeUPC_E0, bcCodeUPC_E1, bcC'
10960:    +'odeUPC_Supp2, bcCodeUPC_Supp5, bcCodeEAN128A, bcCodeEAN128B, bcCodeEAN128C
10961:   TBarLineType', '( white, black, black_half )
10962:   TBarcodeOption', '( bcoNone, bcoCode, bcoTyp, bcoBoth )
10963:   TShowTextPosition', '( stpTopLeft, stpTopRight, stpTopCenter, st'
10964:    +'pBottomLeft, stpBottomRight, stpBottomCenter )
10965:   TCheckSumMethod', '( csmNone, csmModulo10 )
10966:   SIRegister_TAsBarcode(CL);
10967:  Function CheckSumModulo10( const data : string) : string
10968:  Function ConvertMmToPixelsX( const Value : Double) : Integer
10969:  Function ConvertMmToPixelsY( const Value : Double) : Integer
10970:  Function ConvertInchToPixelsX( const Value : Double) : Integer
10971:  Function ConvertInchToPixelsY( const Value : Double) : Integer
10972: end;
10973:
10974: procedure SIRegister_Geometry(CL: TPSPascalCompiler);  //OpenGL
10975: begin
10976:    THomogeneousByteVector', 'array[0..3] of Byte
10977:   THomogeneousWordVector', 'array[0..3] of Word
10978:   THomogeneousIntVector', 'array[0..3] of Integer
10979:   THomogeneousFltVector', 'array[0..3] of single
10980:   THomogeneousDblVector', 'array[0..3] of double
10981:   THomogeneousExtVector', 'array[0..3] of extended
10982:   TAffineByteVector', 'array[0..2] of Byte
10983:   TAffineWordVector', 'array[0..2] of Word
10984:   TAffineIntVector', 'array[0..2] of Integer
10985:   TAffineFltVector', 'array[0..2] of single
10986:   TAffineDblVector', 'array[0..2] of double
10987:   TAffineExtVector', 'array[0..2] of extended
10988:   THomogeneousByteMatrix', 'array[0..3] of THomogeneousByteVector
10989:   THomogeneousWordMatrix', 'array[0..3] of THomogeneousWordVector
10990:   THomogeneousIntMatrix', 'array[0..3] of THomogeneousIntVector
10991:   THomogeneousFltMatrix', 'array[0..3] of THomogeneousFltVector
10992:   THomogeneousDblMatrix', 'array[0..3] of THomogeneousDblVector
10993:   THomogeneousExtMatrix', 'array[0..3] of THomogeneousExtVector
10994:   TAffineByteMatrix', 'array[0..2] of TAffineByteVector
10995:   TAffineWordMatrix', 'array[0..2] of TAffineWordVector
10996:   TAffineIntMatrix', 'array[0..2] of TAffineIntVector
10997:   TAffineFltMatrix', 'array[0..3] of TAffineFltVector
10998:   TAffineDblMatrix', 'array[0..3] of TAffineDblVector
10999:   TAffineExtMatrix', 'array[0..3] of TAffineExtVector
11000:   TMatrix4b', 'THomogeneousByteMatrix
11001:   TMatrix4w', 'THomogeneousWordMatrix
11002:   TMatrix4i', 'THomogeneousIntMatrix
11003:   TMatrix4f', 'THomogeneousFltMatrix
11004:   TMatrix4d', 'THomogeneousDblMatrix
11005:   TMatrix4e', 'THomogeneousExtMatrix
11006:   TMatrix3b', 'TAffineByteMatrix
11007:   TMatrix3w', 'TAffineWordMatrix
11008:   TMatrix3i', 'TAffineIntMatrix
11009:   TMatrix3f', 'TAffineFltMatrix
11010:   TMatrix3d', 'TAffineDblMatrix
11011:   TMatrix3e', 'TAffineExtMatrix
11012:   //'PMatrix', '^TMatrix // will not work
11013:   TMatrixGL', 'THomogeneousFltMatrix
11014:   THomogeneousMatrix', 'THomogeneousFltMatrix
11015:   TAffineMatrix', 'TAffineFltMatrix
11016:   TQuaternion', 'record Vector : TVector4f; end
```

```
11017:   TRectangle', 'record Left : integer; Top : integer; Width : inte'
11018:   +'ger; Height : Integer; end
11019:   TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
11020:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
11021:    +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )
11022:  'EPSILON','Extended').setExtended( 1E-100);
11023:  'EPSILON2','Extended').setExtended( 1E-50);
11024:  Function VectorAddGL( V1, V2 : TVectorGL) : TVectorGL
11025:  Function VectorAffineAdd( V1, V2 : TAffineVector) : TAffineVector
11026:  Function VectorAffineCombine(V1,V2:TAffineVector; F1, F2 : Single) : TAffineVector
11027:  Function VectorAffineDotProduct( V1, V2 : TAffineVector) : Single
11028:  Function VectorAffineLerp( V1, V2 : TAffineVector; t : Single) : TAffineVector
11029:  Function VectorAffineSubtract( V1, V2 : TAffineVector) : TAffineVector
11030:  Function VectorAngle( V1, V2 : TAffineVector) : Single
11031:  Function VectorCombine( V1, V2 : TVectorGL; F1, F2 : Single) : TVectorGL
11032:  Function VectorCrossProduct( V1, V2 : TAffineVector) : TAffineVector
11033:  Function VectorDotProduct( V1, V2 : TVectorGL) : Single
11034:  Function VectorLength( V : array of Single) : Single
11035:  Function VectorLerp( V1, V2 : TVectorGL; t : Single) : TVectorGL
11036:  Procedure VectorNegate( V : array of Single)
11037:  Function VectorNorm( V : array of Single) : Single
11038:  Function VectorNormalize( V : array of Single) : Single
11039:  Function VectorPerpendicular( V, N : TAffineVector) : TAffineVector
11040:  Function VectorReflect( V, N : TAffineVector) : TAffineVector
11041:  Procedure VectorRotate( var Vector : TVector4f; Axis : TVector3f; Angle : Single)
11042:  Procedure VectorScale( V : array of Single; Factor : Single)
11043:  Function VectorSubtractGL( V1, V2 : TVectorGL) : TVectorGL
11044:  Function CreateRotationMatrixX( Sine, Cosine : Single) : TMatrixGL
11045:  Function CreateRotationMatrixY( Sine, Cosine : Single) : TMatrixGL
11046:  Function CreateRotationMatrixZ( Sine, Cosine : Single) : TMatrixGL
11047:  Function CreateScaleMatrix( V : TAffineVector) : TMatrixGL
11048:  Function CreateTranslationMatrix( V : TVectorGL) : TMatrixGL
11049:  Procedure MatrixAdjoint( var M : TMatrixGL)
11050:  Function MatrixAffineDeterminant( M : TAffineMatrix) : Single
11051:  Procedure MatrixAffineTranspose( var M : TAffineMatrix)
11052:  Function MatrixDeterminant( M : TMatrixGL) : Single
11053:  Procedure MatrixInvert( var M : TMatrixGL)
11054:  Function MatrixMultiply( M1, M2 : TMatrixGL) : TMatrixGL
11055:  Procedure MatrixScale( var M : TMatrixGL; Factor : Single)
11056:  Procedure MatrixTranspose( var M : TMatrixGL)
11057:  Function QuaternionConjugate( Q : TQuaternion) : TQuaternion
11058:  Function QuaternionFromPoints( V1, V2 : TAffineVector) : TQuaternion
11059:  Function QuaternionMultiply( qL, qR : TQuaternion) : TQuaternion
11060:  Function QuaternionSlerp( QStart,QEnd:TQuaternion; Spin:Integer; t:Single):TQuaternion
11061:  Function QuaternionToMatrix( Q : TQuaternion) : TMatrixGL
11062:  Procedure QuaternionToPoints( Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
11063:  Function ConvertRotation( Angles : TAffineVector) : TVectorGL
11064:  Function CreateRotationMatrix( Axis : TVector3f; Angle : Single) : TMatrixGL
11065:  //Function MatrixDecompose( M : TMatrixGL; var Tran : TTransformations) : Boolean
11066:  Function VectorAffineTransform( V : TAffineVector; M : TAffineMatrix) : TAffineVector
11067:  Function VectorTransform( V : TVector4f; M : TMatrixGL) : TVector4f;
11068:  Function VectorTransform1( V : TVector3f; M : TMatrixGL) : TVector3f;
11069:  Function MakeAffineDblVector( V : array of Double) : TAffineDblVector
11070:  Function MakeDblVector( V : array of Double) : THomogeneousDblVector
11071:  Function MakeAffineVector( V : array of Single) : TAffineVector
11072:  Function MakeQuaternion( Imag : array of Single; Real : Single) : TQuaternion
11073:  Function MakeVector( V : array of Single) : TVectorGL
11074:  Function PointInPolygonGL( xp, yp : array of Single; x, y : Single) : Boolean
11075:  Function VectorAffineDblToFlt( V : TAffineDblVector) : TAffineVector
11076:  Function VectorDblToFlt( V : THomogeneousDblVector) : THomogeneousVector
11077:  Function VectorAffineFltToDbl( V : TAffineVector) : TAffineDblVector
11078:  Function VectorFltToDbl( V : TVectorGL) : THomogeneousDblVector
11079:  Function ArcCosGL( X : Extended) : Extended
11080:  Function ArcSinGL( X : Extended) : Extended
11081:  Function ArcTan2GL( Y, X : Extended) : Extended
11082:  Function CoTanGL( X : Extended) : Extended
11083:  Function DegToRadGL( Degrees : Extended) : Extended
11084:  Function RadToDegGL( Radians : Extended) : Extended
11085:  Procedure SinCosGL( Theta : Extended; var Sin, Cos : Extended)
11086:  Function TanGL( X : Extended) : Extended
11087:  Function Turn( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11088:  Function Turn1( Matrix : TMatrixGL; MasterUp : TAffineVector; Angle: Single): TMatrixGL;
11089:  Function Pitch( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11090:  Function Pitch1( Matrix : TMatrixGL; MasterRight:TAffineVector;Angle:Single):TMatrixGL;
11091:  Function Roll( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11092:  Function Roll1( Matrix:TMatrixGL; MasterDirection:TAffineVector;Angle:Single):TMatrixGL;
11093: end;
11094:
11095:
11096: procedure SIRegister_JclRegistry(CL: TPSPascalCompiler);
11097: begin
11098:  Function RegCreateKey( const RootKey : HKEY; const Key, Value : string) : Longint
11099:  Function RegDeleteEntry( const RootKey : HKEY; const Key, Name : string) : Boolean
11100:  Function RegDeleteKeyTree( const RootKey : HKEY; const Key : string) : Boolean
11101:  Function RegReadBool( const RootKey : HKEY; const Key, Name : string) : Boolean
11102:  Function RegReadBoolDef( const RootKey : HKEY; const Key, Name : string; Def : Boolean) : Boolean
11103:  Function RegReadInteger( const RootKey : HKEY; const Key, Name : string) : Integer
11104:  Function RegReadIntegerDef( const RootKey : HKEY; const Key, Name : string; Def : Integer) : Integer
11105:  Function RegReadString( const RootKey : HKEY; const Key, Name : string) : string
```

```
11106:  Function RegReadStringDef( const RootKey : HKEY; const Key, Name, Def : string) : string
11107:  Function RegReadDWORD( const RootKey : HKEY; const Key, Name : string) : Int64
11108:  Function RegReadDWORDDef( const RootKey : HKEY; const Key, Name : string; Def : Int64) : Int64
11109:  Procedure RegWriteBool( const RootKey : HKEY; const Key, Name : string; Value : Boolean)
11110:  Procedure RegWriteInteger( const RootKey : HKEY; const Key, Name : string; Value : Integer)
11111:  Procedure RegWriteString( const RootKey : HKEY; const Key, Name, Value : string)
11112:  Procedure RegWriteDWORD( const RootKey : HKEY; const Key, Name : string; Value : Int64)
11113:  Function RegGetValueNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11114:  Function RegGetKeyNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11115:  Function RegHasSubKeys( const RootKey : HKEY; const Key : string) : Boolean
11116:  Function RegKeyExists( const RootKey : HKEY; const Key : string) : Boolean
11117:   AddTypeS('TExecKind', '( ekMachineRun, ekMachineRunOnce, ekUserRun, ekUser'
11118:    +'RunOnce, ekServiceRun, ekServiceRunOnce )
11119:   AddClassN(FindClass('TOBJECT'),'EJclRegistryError
11120:  Function UnregisterAutoExec( ExecKind : TExecKind; const Name : string) : Boolean
11121:  Function RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline : string) : Boolean
11122:  Function RegSaveList(const RootKey:HKEY; const Key:string; const ListName:string;const
       Items:TStrings):Bool;
11123:  Function RegLoadList(const RootKey:HKEY; const Key:string; const ListName:string;const
       SaveTo:TStrings):Bool;
11124:  Function RegDelList( const RootKey:HKEY; const Key:string; const ListName:string): Boolean
11125: end;
11126:
11127: procedure SIRegister_JclCOM(CL: TPSPascalCompiler);
11128: begin
11129:  CLSID_StdComponentCategoriesMgr','TGUID').SetString( '{0002E005-0000-0000-C000-000000000046}
11130:  CATID_SafeForInitializing','TGUID').SetString( '{7DD95802-9882-11CF-9FA9-00AA006C42C4}
11131:  CATID_SafeForScripting','TGUID').SetString( '{7DD95801-9882-11CF-9FA9-00AA006C42C4}
11132:  icMAX_CATEGORY_DESC_LEN','LongInt').SetInt( 128);
11133:   FindClass('TOBJECT'),'EInvalidParam
11134:  Function IsDCOMInstalled : Boolean
11135:  Function IsDCOMEnabled : Boolean
11136:  Function GetDCOMVersion : string
11137:  Function GetMDACVersion : string
11138:  Function MarshalInterThreadInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11139:  Function MarshalInterProcessInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11140:  Function MarshalInterProcessInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11141:  Function MarshalInterMachineInterfaceInStream( const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11142:  Function MarshalInterMachineInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
       VarArray:OleVariant):HResult;
11143:  Function CreateComponentCategory( const CatID : TGUID; const sDescription : string) : HResult
11144:  Function RegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11145:  Function UnRegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11146:  Function ResetIStreamToStart( Stream : IStream) : Boolean
11147:  Function SizeOfIStreamContents( Stream : IStream) : Largeint
11148:  Function StreamToVariantArray( Stream : TStream) : OleVariant;
11149:  Function StreamToVariantArray1( Stream : IStream) : OleVariant;
11150:  Procedure VariantArrayToStream( VarArray : OleVariant; var Stream : TStream);
11151:  Procedure VariantArrayToStream1( VarArray : OleVariant; var Stream : IStream);
11152: end;
11153:
11154:
11155: procedure SIRegister_JclUnitConv_mX2(CL: TPSPascalCompiler);
11156: begin
11157:  Const('CelsiusFreezingPoint','Extended').setExtended( 0.0);
11158:  FahrenheitFreezingPoint','Extended').setExtended( 32.0);
11159:  KelvinFreezingPoint','Extended').setExtended( 273.15);
11160:  CelsiusAbsoluteZero','Extended').setExtended( - 273.15);
11161:  FahrenheitAbsoluteZero','Extended').setExtended( - 459.67);
11162:  KelvinAbsoluteZero','Extended').setExtended( 0.0);
11163:  DegPerCycle','Extended').setExtended( 360.0);
11164:  DegPerGrad','Extended').setExtended( 0.9);
11165:  DegPerRad','Extended').setExtended( 57.295779513082320876798154814105);
11166:  GradPerCycle','Extended').setExtended( 400.0);
11167:  GradPerDeg','Extended').setExtended( 1.11111111111111111111111111111111);
11168:  GradPerRad','Extended').setExtended( 63.661977236758134307553505349006);
11169:  RadPerCycle','Extended').setExtended( 6.2831853071795864769252867665590);
11170:  RadPerDeg','Extended').setExtended( 0.017453292519943295769236907684886);
11171:  RadPerGrad','Extended').setExtended( 0.015707963267948966192313216916398);
11172:  CyclePerDeg','Extended').setExtended( 0.002777777777777777777777777777778);
11173:  CyclePerGrad','Extended').setExtended( 0.0025);
11174:  CyclePerRad','Extended').setExtended( 0.15915494309189533576888376337251);
11175:  ArcMinutesPerDeg','Extended').setExtended( 60.0);
11176:  ArcSecondsPerArcMinute','Extended').setExtended( 60.0);
11177:  Function HowAOneLinerCanBiteYou( const Step, Max : Longint) : Longint
11178:  Function MakePercentage( const Step, Max : Longint) : Longint
11179:  Function CelsiusToKelvin( const T : double) : double
11180:  Function CelsiusToFahrenheit( const T : double) : double
11181:  Function KelvinToCelsius( const T : double) : double
11182:  Function KelvinToFahrenheit( const T : double) : double
11183:  Function FahrenheitToCelsius( const T : double) : double
11184:  Function FahrenheitToKelvin( const T : double) : double
11185:  Function CycleToDeg( const Cycles : double) : double
11186:  Function CycleToGrad( const Cycles : double) : double
11187:  Function CycleToRad( const Cycles : double) : double
11188:  Function DegToCycle( const Degrees : double) : double
11189:  Function DegToGrad( const Degrees : double) : double
```

```
11190:  Function DegToRad( const Degrees : double) : double
11191:  Function GradToCycle( const Grads : double) : double
11192:  Function GradToDeg( const Grads : double) : double
11193:  Function GradToRad( const Grads : double) : double
11194:  Function RadToCycle( const Radians : double) : double
11195:  Function RadToDeg( const Radians : double) : double
11196:  Function RadToGrad( const Radians : double) : double
11197:  Function DmsToDeg( const D, M : Integer; const S : double) : double
11198:  Function DmsToRad( const D, M : Integer; const S : double) : double
11199:  Procedure DegToDms( const Degrees : double; out D, M : Integer; out S : double)
11200:  Function DegToDmsStr( const Degrees : double; const SecondPrecision : Cardinal) : string
11201:  Procedure CartesianToPolar( const X, Y : double; out R, Phi : double)
11202:  Procedure PolarToCartesian( const R, Phi : double; out X, Y : double)
11203:  Procedure CartesianToCylinder( const X, Y, Z : double; out R, Phi, Zeta : double)
11204:  Procedure CartesianToSpheric( const X, Y, Z : double; out Rho, Phi, Theta : double)
11205:  Procedure CylinderToCartesian( const R, Phi, Zeta : double; out X, Y, Z : double)
11206:  Procedure SphericToCartesian( const Rho, Theta, Phi : double; out X, Y, Z : double)
11207:  Function CmToInch( const Cm : double) : double
11208:  Function InchToCm( const Inch : double) : double
11209:  Function FeetToMetre( const Feet : double) : double
11210:  Function MetreToFeet( const Metre : double) : double
11211:  Function YardToMetre( const Yard : double) : double
11212:  Function MetreToYard( const Metre : double) : double
11213:  Function NmToKm( const Nm : double) : double
11214:  Function KmToNm( const Km : double) : double
11215:  Function KmToSm( const Km : double) : double
11216:  Function SmToKm( const Sm : double) : double
11217:  Function LitreToGalUs( const Litre : double) : double
11218:  Function GalUsToLitre( const GalUs : double) : double
11219:  Function GalUsToGalCan( const GalUs : double) : double
11220:  Function GalCanToGalUs( const GalCan : double) : double
11221:  Function GalUsToGalUk( const GalUs : double) : double
11222:  Function GalUkToGalUs( const GalUk : double) : double
11223:  Function LitreToGalCan( const Litre : double) : double
11224:  Function GalCanToLitre( const GalCan : double) : double
11225:  Function LitreToGalUk( const Litre : double) : double
11226:  Function GalUkToLitre( const GalUk : double) : double
11227:  Function KgToLb( const Kg : double) : double
11228:  Function LbToKg( const Lb : double) : double
11229:  Function KgToOz( const Kg : double) : double
11230:  Function OzToKg( const Oz : double) : double
11231:  Function CwtUsToKg( const Cwt : double) : double
11232:  Function CwtUkToKg( const Cwt : double) : double
11233:  Function KaratToKg( const Karat : double) : double
11234:  Function KgToCwtUs( const Kg : double) : double
11235:  Function KgToCwtUk( const Kg : double) : double
11236:  Function KgToKarat( const Kg : double) : double
11237:  Function KgToSton( const Kg : double) : double
11238:  Function KgToLton( const Kg : double) : double
11239:  Function StonToKg( const STon : double) : double
11240:  Function LtonToKg( const Lton : double) : double
11241:  Function QrUsToKg( const Qr : double) : double
11242:  Function QrUkToKg( const Qr : double) : double
11243:  Function KgToQrUs( const Kg : double) : double
11244:  Function KgToQrUk( const Kg : double) : double
11245:  Function PascalToBar( const Pa : double) : double
11246:  Function PascalToAt( const Pa : double) : double
11247:  Function PascalToTorr( const Pa : double) : double
11248:  Function BarToPascal( const Bar : double) : double
11249:  Function AtToPascal( const At : double) : double
11250:  Function TorrToPascal( const Torr : double) : double
11251:  Function KnotToMs( const Knot : double) : double
11252:  Function HpElectricToWatt( const HpE : double) : double
11253:  Function HpMetricToWatt( const HpM : double) : double
11254:  Function MsToKnot( const ms : double) : double
11255:  Function WattToHpElectric( const W : double) : double
11256:  Function WattToHpMetric( const W : double) : double
11257:  function getBigPI: string;    //PI of 1000 numbers
11258:
11259: procedure SIRegister_devcutils(CL: TPSPascalCompiler);
11260: begin
11261:  Function CDExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
11262:  Procedure CDCopyFile( const FileName, DestName : string)
11263:  Procedure CDMoveFile( const FileName, DestName : string)
11264:  Function MakeCommaTextToColor( Text : string; Index : Integer; DefaultColor : TColor) : TColor
11265:  Procedure CDDeleteFiles( Sender : TObject; s : string)
11266:  Function CDGetTempDir : string
11267:  Function CDGetFileSize( FileName : string) : longint
11268:  Function GetFileTime( FileName : string) : longint
11269:  Function GetShortName( FileName : string) : string
11270:  Function GetFullName( FileName : string) : string
11271:  Function WinReboot : boolean
11272:  Function WinDir : String
11273:  Function RunFile( FileToRun : string; Params : string; Dir : string; Wait : boolean) : cardinal
11274:  Function RunFile_( Cmd, WorkDir : string; Wait : boolean) : Boolean
11275:  Function devExecutor : TdevExecutor
11276: end;
11277:
11278: procedure SIRegister_FileAssocs(CL: TPSPascalCompiler);
```

```
11279: begin
11280:  Procedure CheckAssociations // AssociationsCount','LongInt').SetInt( 7);
11281:  Procedure Associate( Index : integer)
11282:  Procedure UnAssociate( Index : integer)
11283:  Function IsAssociated( Index : integer) : boolean
11284:  Function CheckFiletype( const extension, filetype, description, verb, serverapp : string) : boolean
11285:  Procedure RegisterFiletype( const extension, filetype, description, verb, serverapp,IcoNum: string)
11286:  Procedure RegisterDDEServer( const filetype, verb, topic, servername, macro : string)
11287:  procedure RefreshIcons;
11288: function GetShadeColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11289: function MergColor(Colors: Array of TColor): TColor;
11290: function NewColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11291: procedure DimBitmap(ABitmap: TBitmap; Value: integer);
11292: function GrayColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11293: function GetInverseColor(AColor: TColor): TColor;
11294: procedure GrayBitmap(ABitmap: TBitmap; Value: integer);
11295: procedure DrawBitmapShadow(B: TBitmap; ACanvas: TCanvas;X,Y: integer;ShadowColor: TColor);
11296: procedure DrawCheckMark(ACanvas: TCanvas; X, Y: integer);
11297: Procedure GetSystemMenuFont(Font: TFont);
11298: end;
11299:
11300: //*************************unit uPSI_JvHLParser;*****************************
11301: function IsStringConstant(const St: string): Boolean;
11302: function IsIntConstant(const St: string): Boolean;
11303: function IsRealConstant(const St: string): Boolean;
11304: function IsIdentifier(const ID: string): Boolean;
11305: function GetStringValue(const St: string): string;
11306: procedure ParseString(const S: string; Ss: TStrings);
11307: function IsStringConstantW(const St: WideString): Boolean;
11308: function IsIntConstantW(const St: WideString): Boolean;
11309: function IsRealConstantW(const St: WideString): Boolean;
11310: function IsIdentifierW(const ID: WideString): Boolean;
11311: function GetStringValueW(const St: WideString): WideString;
11312: procedure ParseStringW(const S: WideString; Ss: TStrings);
11313:
11314:
11315: //*************************unit uPSI_JclMapi;*****************************
11316:
11317: Function JclSimpleSendMail( const ARecipient,AName,ASubject, ABody : string; const AAttachment :
       TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11318: Function JclSimpleSendFax( const ARecipient, AName,ASubject, ABody : string; const AAttachment :
       TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11319: Function JclSimpleBringUpSendMailDialog(const ASubject,ABody:string;const
       AAttach:TFileName;AParentWND:HWND):Bool;
11320: Function MapiCheck( const Res : DWORD; IgnoreUserAbort : Boolean) : DWORD
11321: Function MapiErrorMessage( const ErrorCode : DWORD) : string
11322:
11323: procedure SIRegister_IdNTLM(CL: TPSPascalCompiler);
11324: begin
11325:   //'Pdes_key_schedule', '^des_key_schedule // will not work
11326:  Function BuildType1Message( ADomain, AHost : String) : String
11327:  Function BuildType3Message(ADomain,AHost,AUsername:WideString;APassword,ANonce:String):String
11328:  Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
11329:  Function FindAuthClass( AuthName : String) : TIdAuthenticationClass
11330: GBase64CodeTable','string').SetString('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
11331: GXXECodeTable','string').SetString('+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
11332: GUUECodeTable','string').SetString('`!"#$%&''()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
11333: end;
11334:
11335: procedure SIRegister_WDosSocketUtils(CL: TPSPascalCompiler);
11336: begin
11337: ('IpAny','LongWord').SetUInt( $00000000);
11338:  IpLoopBack','LongWord').SetUInt( $7F000001);
11339:  IpBroadcast','LongWord').SetUInt( $FFFFFFFF);
11340:  IpNone','LongWord').SetUInt( $FFFFFFFF);
11341:  PortAny','LongWord').SetUInt( $0000);
11342:  SocketMaxConnections','LongInt').SetInt( 5);
11343:  TIpAddr', 'LongWord
11344:  TIpRec', 'record IpB1 : byte; IpB2 : byte; IpB3 : byte; IpB4 : Byte; end
11345:  Function HostToNetLong( HostLong : LongWord) : LongWord
11346:  Function HostToNetShort( HostShort : Word) : Word
11347:  Function NetToHostLong( NetLong : LongWord) : LongWord
11348:  Function NetToHostShort( NetShort : Word) : Word
11349:  Function StrToIp( Ip : string) : TIpAddr
11350:  Function IpToStr( Ip : TIpAddr) : string
11351: end;
11352:
11353: (*----------------------------------------------------------------------------*)
11354: procedure SIRegister_ALSMTPClient(CL: TPSPascalCompiler);
11355: begin
11356:   TAlSmtpClientAuthType', '( AlsmtpClientAuthNone, alsmtpClientAut'
11357:    +'hPlain, AlsmtpClientAuthLogin, AlsmtpClientAuthCramMD5, AlsmtpClientAuthCr'
11358:    +'amSha1, AlsmtpClientAuthAutoSelect )
11359:   TAlSmtpClientAuthTypeSet', 'set of TAlSmtpClientAuthType
11360:   SIRegister_TAlSmtpClient(CL);
11361: end;
11362:
11363: procedure SIRegister_WDosPlcUtils(CL: TPSPascalCompiler);
11364: begin
```

```
11365: 'TBitNo', 'Integer
11366:   TStByteNo', 'Integer
11367: TStationNo', 'Integer
11368: TInOutNo', 'Integer
11369: TIo', '( EE, AA, NE, NA )
11370: TBitSet', 'set of TBitNo
11371: TAddrKind', 'set of ( akBit0, akBit1, akBit2, akOut, akNot, akBus )
11372: TBitAddrRec', 'record Kind : TAddrKind; InOutNo : TInOutNo; ByteNo : Byte; end
11373: TBitAddr', 'LongInt
11374: TByteAddrRec', 'record Kind : TAddrKind; ByteNo : Byte; end
11375: TByteAddr', 'SmallInt
11376: TInOutState', '( iosInit, iosHalt, iosRun, iosError )
11377:  Function BitAddr(aIo: TIo; aInOutNo : TInOutNo; aByteNo : Byte; aBitNo : TBitNo) : TBitAddr
11378:  Function BusBitAddr(aIo:TIo;aInOutNo:TInOutNo;aStat:TStationNo;aStByteNo:TStByteNo;aBitNo:TBitNo) :
       TBitAddr;
11379:  Procedure BitAddrToValues(aBitAdr:TBitAdr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte;var
       aBitNo:TBitNo);
11380:  Function BitAddrToStr( Value : TBitAddr) : string
11381:  Function StrToBitAddr( const Value : string) : TBitAddr
11382:  Function ByteAddr( aIo : TIo; aInOutNo : TInOutNo; aByteNo : Byte) : TByteAddr
11383:  Function BusByteAddr(aIo:TIo;aInOutNo:TInOutNo;aStation:TStationNo;aStByteNo: TStByteNo):TByteAddr
11384:  Procedure ByteAddrToValues(aByteAddr:TByteAddr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte)
11385:  Function ByteAddrToStr( Value : TByteAddr) : string
11386:  Function StrToByteAddr( const Value : string) : TByteAddr
11387:  Procedure IncByteAddr( var ByteAddr : TByteAddr; Increment : Integer)
11388:  Procedure DecByteAddr( var ByteAddr : TByteAddr; Decrement : Integer)
11389:  Function InOutStateToStr( State : TInOutState) : string
11390:  Function MasterErrorToStr( ErrorCode : TErrorCode) : string
11391:  Function SlaveErrorToStr( ErrorCode : TErrorCode) : string
11392: end;
11393:
11394: procedure SIRegister_WDosTimers(CL: TPSPascalCompiler);
11395: begin
11396: TIntFreq', '( ifNone, if32768, if16384, if8192, if4096, if2048, '
11397:    +'if1024, if512, if256, if128, if64, if32, if16, if8, if4, if2 )
11398: DpmiPmVector', 'Int64
11399:  'DInterval','LongInt').SetInt( 1000);
11400:  //'DEnabled','Boolean')BoolToStr( True);
11401:  'DIntFreq','string').SetString(' if64
11402:  //'DMessages','Boolean').SetString( if64);
11403:   SIRegister_TwdxCustomTimer(CL);
11404:   SIRegister_TwdxTimer(CL);
11405:   SIRegister_TwdxRtcTimer(CL);
11406:   SIRegister_TCustomIntTimer(CL);
11407:   SIRegister_TIntTimer(CL);
11408:   SIRegister_TRtcIntTimer(CL);
11409:  Function RealNow : TDateTime
11410:  Function MsToDateTime( MilliSecond : LongInt) : TDateTime
11411:  Function DateTimeToMs( Time : TDateTime) : LongInt
11412: end;
11413:
11414: procedure SIRegister_IdSysLogMessage(CL: TPSPascalCompiler);
11415: begin
11416: TIdSyslogPRI', 'Integer
11417: TIdSyslogFacility', '( sfKernel, sfUserLevel, sfMailSystem, sfSy'
11418:    +'stemDaemon, sfSecurityOne, sfSysLogInternal, sfLPR, sfNNTP, sfUUCP, sfCloc'
11419:    +'kDaemonOne, sfSecurityTwo, sfFTPDaemon, sfNTP, sfLogAudit, sfLogAlert, sfC'
11420:    +'lockDaemonTwo, sfLocalUseZero, sfLocalUseOne, sfLocalUseTwo, sfLocalUseThr'
11421:    +'ee, sfLocalUseFour, sfLocalUseFive, sfLocalUseSix, sfLocalUseSeven )
11422: TIdSyslogSeverity', '( slEmergency, slAlert, slCritical, slError'
11423:    +', slWarning, slNotice, slInformational, slDebug )
11424:   SIRegister_TIdSysLogMsgPart(CL);
11425:   SIRegister_TIdSysLogMessage(CL);
11426:  Function FacilityToString( AFac : TIdSyslogFacility) : string
11427:  Function SeverityToString( ASec : TIdsyslogSeverity) : string
11428:  Function NoToSeverity( ASev : Word) : TIdSyslogSeverity
11429:  Function logSeverityToNo( ASev : TIdSyslogSeverity) : Word
11430:  Function NoToFacility( AFac : Word) : TIdSyslogFacility
11431:  Function logFacilityToNo( AFac : TIdSyslogFacility) : Word
11432: end;
11433:
11434: procedure SIRegister_TextUtils(CL: TPSPascalCompiler);
11435: begin
11436:  'UWhitespace','String').SetString( '(?:\s*)
11437:  Function StripSpaces( const AText : string) : string
11438:  Function CharCount( const AText : string; Ch : Char) : Integer
11439:  Function BalancedText( const AText : string; const Ch1, Ch2 : Char; const Count : Integer) : string
11440:  Function BalancedTextReg( const AText:string; const Ch1, Ch2 : Char; const Count : Integer) : string
11441: end;
11442:
11443:
11444: procedure SIRegister_ExtPascalUtils(CL: TPSPascalCompiler);
11445: begin
11446:  ExtPascalVersion','String').SetString( '0.9.8
11447:   AddTypeS('TBrowser', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
11448:    +'Opera, brKonqueror, brMobileSafari )
11449:   AddTypeS('TCSSUnit', '( cssPX, cssPerc, cssEM, cssEX, cssIN, cssCM, cssMM, cssPT, cssPC, cssnone )
11450:   AddTypeS('TExtProcedure', 'Procedure
11451:  Function DetermineBrowser( const UserAgentStr : string) : TBrowser
```

```
11452:  Function ExtExtract(const Delims:array of string;var S:string;var Matches:TStringList;Remove:bool):bool;
11453:  Function ExtExplode( Delim : char; const S : string; Separator : char) : TStringList
11454:  Function FirstDelimiter( const Delimiters, S : string; Offset : integer) : integer
11455:  Function RPosEx( const Substr, Str : string; Offset : integer) : integer
11456:  Function CountStr( const Substr, Str : string; UntilStr : string) : integer
11457:  Function StrToJS( const S : string; UseBR : boolean) : string
11458:  Function CaseOf( const S : string; const Cases : array of string) : integer
11459:  Function RCaseOf( const S : string; const Cases : array of string) : integer
11460:  Function EnumToJSString( TypeInfo : PTypeInfo; Value : integer) : string
11461:  Function SetPaddings(Top:integer;Right:int;Bottom:intr;Left:integer;CSSUnit:TCSSUnit;Header:bool):string;
11462:  Function SetMargins(Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool): string;
11463:  Function ExtBefore( const BeforeS, AfterS, S : string) : boolean
11464:  Function IsUpperCase( S : string) : boolean
11465:  Function BeautifyJS(const AScript:string;const StartingLevel:integer;SplitHTMLNewLine: boolean):string;
11466:  Function BeautifyCSS( const AStyle : string) : string
11467:  Function LengthRegExp( Rex : string; CountAll : Boolean) : integer
11468:  Function JSDateToDateTime( JSDate : string) : TDateTime
11469: end;
11470:
11471: procedure SIRegister_JclShell(CL: TPSPascalCompiler);
11472: begin
11473:   TSHDeleteOption', '( doSilent, doAllowUndo, doFilesOnly )
11474:   TSHDeleteOptions', 'set of TSHDeleteOption
11475:   TSHRenameOption', '( roSilent, roRenameOnCollision )
11476:   TSHRenameOptions', 'set of TSHRenameOption
11477:  Function SHDeleteFiles( Parent : HWND; const Files : string; Options : TSHDeleteOptions) : Boolean
11478:  Function SHDeleteFolder( Parent : HWND; const Folder : string; Options : TSHDeleteOptions) : Boolean
11479:  Function SHRenameFile( const Src, Dest : string; Options : TSHRenameOptions) : Boolean
11480:   TEnumFolderFlag', '( efFolders, efNonFolders, efIncludeHidden )
11481:   TEnumFolderFlags', 'set of TEnumFolderFlag
11482:   TEnumFolderRec', 'record DisplayName : string; Attributes : DWOR'
11483:    +'D; IconLarge : HICON; IconSmall : HICON; Item : PItemIdList; EnumIdList : '
11484:    +'IEnumIdList; Folder : IShellFolder; end
11485:  Function SHEnumFolderFirst(const Folder:string;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Boolean;
11486:  Function SHEnumSpecialFolderFirst(SpecialFolder:DWORD;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11487:  Procedure SHEnumFolderClose( var F : TEnumFolderRec)
11488:  Function SHEnumFolderNext( var F : TEnumFolderRec) : Boolean
11489:  Function GetSpecialFolderLocation( const Folder : Integer) : string
11490:  Function DisplayPropDialog( const Handle : HWND; const FileName : string) : Boolean;
11491:  Function DisplayPropDialog1( const Handle : HWND; const Item : PItemIdList) : Boolean;
11492:  Function DisplayContextMenu( const Handle : HWND; const FileName : string; Pos : TPoint) : Boolean
11493:  Function OpenFolder( const Path : string; Parent : HWND) : Boolean
11494:  Function OpenSpecialFolder( FolderID : Integer; Parent : HWND) : Boolean
11495:  Function SHReallocMem( var P : Pointer; Count : Integer) : Boolean
11496:  Function SHAllocMem( out P : Pointer; Count : Integer) : Boolean
11497:  Function SHGetMem( var P : Pointer; Count : Integer) : Boolean
11498:  Function SHFreeMem( var P : Pointer) : Boolean
11499:  Function DriveToPidlBind( const DriveName : string; out Folder : IShellFolder) : PItemIdList
11500:  Function PathToPidl( const Path : string; Folder : IShellFolder) : PItemIdList
11501:  Function PathToPidlBind( const FileName : string; out Folder : IShellFolder) : PItemIdList
11502:  Function PidlBindToParent(const IdList:PItemIdList;out Folder:IShellFolder;out Last:PItemIdList):Bool;
11503:  Function PidlCompare( const Pidl1, Pidl2 : PItemIdList) : Boolean
11504:  Function PidlCopy( const Source : PItemIdList; out Dest : PItemIdList) : Boolean
11505:  Function PidlFree( var IdList : PItemIdList) : Boolean
11506:  Function PidlGetDepth( const Pidl : PItemIdList) : Integer
11507:  Function PidlGetLength( const Pidl : PItemIdList) : Integer
11508:  Function PidlGetNext( const Pidl : PItemIdList) : PItemIdList
11509:  Function PidlToPath( IdList : PItemIdList) : string
11510:  Function StrRetFreeMem( StrRet : TStrRet) : Boolean
11511:  Function StrRetToString( IdList : PItemIdList; StrRet : TStrRet; Free : Boolean) : string
11512:   PShellLink', '^TShellLink // will not work
11513:   TShellLink', 'record Arguments : string; ShowCmd : Integer; Work'
11514:    +'ingDirectory : string; IdList : PItemIDList; Target : string; Description '
11515:    +': string; IconLocation : string; IconIndex : Integer; HotKey : Word; end
11516:  Procedure ShellLinkFree( var Link : TShellLink)
11517:  Function ShellLinkResolve( const FileName : string; var Link : TShellLink) : HRESULT
11518:  Function ShellLinkCreate( const Link : TShellLink; const FileName : string) : HRESULT
11519:  Function ShellLinkCreateSystem(const Link:TShellLink;const Folder:Integer; const FileName:string):HRESULT;
11520:  Function ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon) : Boolean
11521:  Function SHDllGetVersion( const FileName : string; var Version : TDllVersionInfo) : Boolean
11522:  Function GetSystemIcon( IconIndex : Integer; Flags : Cardinal) : HICON
11523:  Function OverlayIcon( var Icon : HICON; Overlay : HICON; Large : Boolean) : Boolean
11524:  Function OverlayIconShortCut( var Large, Small : HICON) : Boolean
11525:  Function OverlayIconShared( var Large, Small : HICON) : Boolean
11526:  Function SHGetItemInfoTip( const Folder : IShellFolder; Item : PItemIdList) : string
11527:  Function ShellExecEx(const FileName:string;const Parameters:string;const Verb:string; CmdShow:Int):Bool;
11528:  Function ShellExec(Wnd: Integer;const Operation,FileName,Parameters,Directy:string;ShowCommand:Int):Bool;
11529:  Function ShellExecAndWait(const FileName:string;const Paramets:string;const Verb:string;CmdShow:Int):Bool;
11530:  Function ShellOpenAs( const FileName : string) : Boolean
11531:  Function ShellRasDial( const EntryName : string) : Boolean
11532:  Function ShellRunControlPanel( const NameOrFileName:string; AppletNumber:Integer):Boolean
11533:  Function GetFileNameIcon( const FileName : string; Flags : Cardinal) : HICON
11534:   TJclFileExeType', '( etError, etMsDos, etWin16, etWin32Gui, etWin32Con )
11535:  Function GetFileExeType( const FileName : TFileName) : TJclFileExeType
11536:  Function ShellFindExecutable( const FileName, DefaultDir : string) : string
11537:  Procedure keybd_event( bVk : Byte; bScan : Byte; dwFlags, dwExtraInfo : DWORD)
11538:  Function OemKeyScan( wOemChar : Word) : DWORD
11539:  Procedure mouse_event( dwFlags, dx, dy, dwData, dwExtraInfo : DWORD)
11540: end;
```

```
11541:
11542: procedure SIRegister_cXMLFunctions(CL: TPSPascalCompiler);
11543: begin
11544:  xmlVersion','String').SetString( '1.0 FindClass('TOBJECT'),'Exml
11545:  //Function xmlValidChar( const Ch : AnsiChar) : Boolean;
11546:  Function xmlValidChar1( const Ch : UCS4Char) : Boolean;
11547:  Function xmlValidChar2( const Ch : WideChar) : Boolean;
11548:  Function xmlIsSpaceChar( const Ch : WideChar) : Boolean
11549:  Function xmlIsLetter( const Ch : WideChar) : Boolean
11550:  Function xmlIsDigit( const Ch : WideChar) : Boolean
11551:  Function xmlIsNameStartChar( const Ch : WideChar) : Boolean
11552:  Function xmlIsNameChar( const Ch : WideChar) : Boolean
11553:  Function xmlIsPubidChar( const Ch : WideChar) : Boolean
11554:  Function xmlValidName( const Text : UnicodeString) : Boolean
11555:  //xmlSpace','Char').SetString( #$20 or  #$9 or  #$D or  #$A);
11556:  //Function xmlSkipSpace( var P : PWideChar) : Boolean
11557:  //Function xmlSkipEq( var P : PWideChar) : Boolean
11558:  //Function xmlExtractQuotedText( var P : PWideChar; var S : UnicodeString) : Boolean
11559:  //Function xmlGetEntityEncoding( const Buf : Pointer; const BufSize : Integer; out HeaderSize : Integer)
      : TUnicodeCodecClass
11560:  Function xmlResolveEntityReference( const RefName : UnicodeString) : WideChar
11561:  Function xmlTag( const Tag : UnicodeString) : UnicodeString
11562:  Function xmlEndTag( const Tag : UnicodeString) : UnicodeString
11563:  Function xmlAttrTag( const Tag : UnicodeString; const Attr : UnicodeString) : UnicodeString
11564:  Function xmlEmptyTag( const Tag, Attr : UnicodeString) : UnicodeString
11565:  Procedure xmlSafeTextInPlace( var Txt : UnicodeString)
11566:  Function xmlSafeText( const Txt : UnicodeString) : UnicodeString
11567:  Function xmlSpaceIndent( const IndentLength : Integer; const IndentLevel : Integer):UnicodeString
11568:  Function xmlTabIndent( const IndentLevel : Integer) : UnicodeString
11569:  Function xmlComment( const Comment : UnicodeString) : UnicodeString
11570:  Procedure SelfTestcXMLFunctions
11571: end;
11572:
11573: (*----------------------------------------------------------------------------*)
11574: procedure SIRegister_DepWalkUtils(CL: TPSPascalCompiler);
11575: begin
11576:  Function AWaitCursor : IUnknown
11577:  Function ChangeCursor( NewCursor : TCursor) : IUnknown
11578:  Procedure SuspendRedraw( AControl : TWinControl; Suspend : boolean)
11579:  Function YesNo( const ACaption, AMsg : string) : boolean
11580:  Procedure strTokenize( const S : string; Delims : TSysCharSet; Results : TStrings)
11581:  Function GetBorlandLibPath( Version : integer; ForDelphi : boolean) : string
11582:  Function GetExpandedLibRoot( Version : integer; ForDelphi : boolean) : string
11583:  Procedure GetPathList( Version : integer; ForDelphi : boolean; Strings : TStrings)
11584:  Procedure GetSystemPaths( Strings : TStrings)
11585:  Procedure MakeEditNumeric( EditHandle : integer)
11586: end;
11587:
11588: procedure SIRegister_yuvconverts(CL: TPSPascalCompiler);
11589: begin
11590:   AddTypeS('TVideoCodec','(vcUnknown,vcRGB,vcYUY2,vcUYVY,vcBTYUV,vcYV,U9,vcYUV12,vcY8,vcY211)
11591:  'BI_YUY2','LongWord').SetUInt( $32595559);
11592:  'BI_UYVY','LongWord').SetUInt( $59565955);
11593:  'BI_BTYUV','LongWord').SetUInt( $50313459);
11594:  'BI_YVU9','LongWord').SetUInt( $39555659);
11595:  'BI_YUV12','LongWord').SetUInt( $30323449);
11596:  'BI_Y8','LongWord').SetUInt( $20203859);
11597:  'BI_Y211','LongWord').SetUInt( $31313259);
11598:  Function BICompressionToVideoCodec( Value : DWord) : TVideoCodec
11599:  Function ConvertCodecToRGB(Codec:TVideoCodec;Src,Dst:Pointer;AWidth,AHeight:Integer):Boolean;
11600: end;
11601:
11602: (*----------------------------------------------------------------------------*)
11603: procedure SIRegister_AviCap(CL: TPSPascalCompiler);
11604: begin
11605:  'WM_USER','LongWord').SetUInt( $0400);
11606:  'WM_CAP_START','LongWord').SetUint($0400);
11607:  'WM_CAP_END','longword').SetUint($0400+85);
11608:  //WM_CAP_START+  85
11609:  //    WM_CAP_SET_CALLBACK_CAPCONTROL  = (WM_CAP_START+  85);
11610:  Function capSetCallbackOnError( hwnd : THandle; fpProc : LongInt) : LongInt
11611:  Function capSetCallbackOnStatus( hwnd : THandle; fpProc : LongInt) : LongInt
11612:  Function capSetCallbackOnYield( hwnd : THandle; fpProc : LongInt) : LongInt
11613:  Function capSetCallbackOnFrame( hwnd : THandle; fpProc : LongInt) : LongInt
11614:  Function capSetCallbackOnVideoStream( hwnd : THandle; fpProc : LongInt) : LongInt
11615:  Function capSetCallbackOnWaveStream( hwnd : THandle; fpProc : LongInt) : LongInt
11616:  Function capSetCallbackOnCapControl( hwnd : THandle; fpProc : LongInt) : LongInt
11617:  Function capSetUserData( hwnd : THandle; lUser : LongInt) : LongInt
11618:  Function capGetUserData( hwnd : THandle) : LongInt
11619:  Function capDriverConnect( hwnd : THandle; I : Word) : LongInt
11620:  Function capDriverDisconnect( hwnd : THandle) : LongInt
11621:  Function capDriverGetName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11622:  Function capDriverGetVersion( hwnd : THandle; szVer : LongInt; wSize : Word) : LongInt
11623:  Function capDriverGetCaps( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11624:  Function capFileSetCaptureFile( hwnd : THandle; szName : LongInt) : LongInt
11625:  Function capFileGetCaptureFile( hwnd : THandle; szName : LongInt; wSize : Word):LongInt
11626:  Function capFileAlloc( hwnd : THandle; dwSize : LongInt) : LongInt
11627:  Function capFileSaveAs( hwnd : THandle; szName : LongInt) : LongInt
11628:  Function capFileSetInfoChunk( hwnd : THandle; lpInfoChunk : LongInt) : LongInt
```

```
11629:  Function capFileSaveDIB( hwnd : THandle; szName : LongInt) : LongInt
11630:  Function capEditCopy( hwnd : THandle) : LongInt
11631:  Function capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11632:  Function capGetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11633:  Function capGetAudioFormatSize( hwnd : THandle) : LongInt
11634:  Function capDlgVideoFormat( hwnd : THandle) : LongInt
11635:  Function capDlgVideoSource( hwnd : THandle) : LongInt
11636:  Function capDlgVideoDisplay( hwnd : THandle) : LongInt
11637:  Function capDlgVideoCompression( hwnd : THandle) : LongInt
11638:  Function capGetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11639:  Function capGetVideoFormatSize( hwnd : THandle) : LongInt
11640:  Function capSetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11641:  Function capPreview( hwnd : THandle; f : Word) : LongInt
11642:  Function capPreviewRate( hwnd : THandle; wMS : Word) : LongInt
11643:  Function capOverlay( hwnd : THandle; f : Word) : LongInt
11644:  Function capPreviewScale( hwnd : THandle; f : Word) : LongInt
11645:  Function capGetStatus( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11646:  Function capSetScrollPos( hwnd : THandle; lpP : LongInt) : LongInt
11647:  Function capGrabFrame( hwnd : THandle) : LongInt
11648:  Function capGrabFrameNoStop( hwnd : THandle) : LongInt
11649:  Function capCaptureSequence( hwnd : THandle) : LongInt
11650:  Function capCaptureSequenceNoFile( hwnd : THandle) : LongInt
11651:  Function capCaptureStop( hwnd : THandle) : LongInt
11652:  Function capCaptureAbort( hwnd : THandle) : LongInt
11653:  Function capCaptureSingleFrameOpen( hwnd : THandle) : LongInt
11654:  Function capCaptureSingleFrameClose( hwnd : THandle) : LongInt
11655:  Function capCaptureSingleFrame( hwnd : THandle) : LongInt
11656:  Function capCaptureGetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11657:  Function capCaptureSetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11658:  Function capSetMCIDeviceName( hwnd : THandle; szName : LongInt) : LongInt
11659:  Function capGetMCIDeviceName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11660:  Function capPaletteOpen( hwnd : THandle; szName : LongInt) : LongInt
11661:  Function capPaletteSave( hwnd : THandle; szName : LongInt) : LongInt
11662:  Function capPalettePaste( hwnd : THandle) : LongInt
11663:  Function capPaletteAuto( hwnd : THandle; iFrames : Word; iColors : LongInt) : LongInt
11664:  Function capPaletteManual( hwnd : THandle; fGrab : Word; iColors : LongInt) : LongInt
11665:   //PCapDriverCaps', '^TCapDriverCaps // will not work
11666:   TCapDriverCaps', 'record wDeviceIndex : WORD; fHasOverlay : BOOL'
11667:    +'; fHasDlgVideoSource : BOOL; fHasDlgVideoFormat : BOOL; fHasDlgVideoDispla'
11668:    +'y : BOOL; fCaptureInitialized : BOOL; fDriverSuppliesPalettes : BOOL; hVid'
11669:    +'eoIn : THANDLE; hVideoOut : THANDLE; hVideoExtIn:THANDLE; hVideoExtOut:THANDLE; end
11670:   //PCapStatus', '^TCapStatus // will not work
11671:   TCapStatus', 'record uiImageWidth : UINT; uiImageHeight : UINT; '
11672:    +'fLiveWindow : BOOL; fOverlayWindow : BOOL; fScale : BOOL; ptScroll : TPOIN'
11673:    +'T; fUsingDefaultPalette : BOOL; fAudioHardware : BOOL; fCapFileExists : BO'
11674:    +'OL; dwCurrentVideoFrame : DWORD; dwCurrentVideoFramesDropped : DWORD; dwCu'
11675:    +'rrentWaveSamples : DWORD; dwCurrentTimeElapsedMS : DWORD; hPalCurrent : HP'
11676:    +'ALETTE; fCapturingNow : BOOL; dwReturn : DWORD; wNumVideoAllocated : WORD;'
11677:    +' wNumAudioAllocated : WORD; end
11678:   //PCaptureParms', '^TCaptureParms // will not work
11679:   TCaptureParms', 'record dwRequestMicroSecPerFrame : DWORD; fMake'
11680:    +'UserHitOKToCapture : BOOL; wPercentDropForError : WORD; fYield : BOOL; dwI'
11681:    +'ndexSize : DWORD; wChunkGranularity : WORD; fUsingDOSMemory : BOOL; wNumVi'
11682:    +'deoRequested : WORD; fCaptureAudio : BOOL; wNumAudioRequested : WORD; vKey'
11683:    +'Abort : WORD; fAbortLeftMouse : BOOL; fAbortRightMouse : BOOL; fLimitEnabl'
11684:    +'ed : BOOL; wTimeLimit : WORD; fMCIControl : BOOL; fStepMCIDevice : BOOL; d'
11685:    +'wMCIStartTime : DWORD; dwMCIStopTime : DWORD; fStepCaptureAt2x : BOOL; wSt'
11686:    +'epCaptureAverageFrames : WORD; dwAudioBufferSize : DWORD; fDisableWriteCac'
11687:    +'he : BOOL; AVStreamMaster : WORD; end
11688:  // PCapInfoChunk', '^TCapInfoChunk // will not work
11689:   //TCapInfoChunk', 'record fccInfoID : FOURCC; lpData : LongInt; cbData : LongInt; end
11690:  'CONTROLCALLBACK_PREROLL','LongInt').SetInt( 1);
11691:  'CONTROLCALLBACK_CAPTURING','LongInt').SetInt( 2);
11692:  Function capCreateCaptureWindow( lpszWindowName: PChar; dwStyle : DWord; x, y : Integer; nWidth, nHeight
       : Integer; hwndParent : THandle; nID : Integer) : THandle
11693:  Function
      capGetDriverDescription(wDriverIndex:DWord;lpszName:PChar;cbName:Integer;lpszVer:PChar;cbVer:Int):Bool;
11694:  'IDS_CAP_BEGIN','LongInt').SetInt( 300);
11695:  'IDS_CAP_END','LongInt').SetInt( 301);
11696:  'IDS_CAP_INFO','LongInt').SetInt( 401);
11697:  'IDS_CAP_OUTOFMEM','LongInt').SetInt( 402);
11698:  'IDS_CAP_FILEEXISTS','LongInt').SetInt( 403);
11699:  'IDS_CAP_ERRORPALOPEN','LongInt').SetInt( 404);
11700:  'IDS_CAP_ERRORPALSAVE','LongInt').SetInt( 405);
11701:  'IDS_CAP_ERRORDIBSAVE','LongInt').SetInt( 406);
11702:  'IDS_CAP_DEFAVIEXT','LongInt').SetInt( 407);
11703:  'IDS_CAP_DEFPALEXT','LongInt').SetInt( 408);
11704:  'IDS_CAP_CANTOPEN','LongInt').SetInt( 409);
11705:  'IDS_CAP_SEQ_MSGSTART','LongInt').SetInt( 410);
11706:  'IDS_CAP_SEQ_MSGSTOP','LongInt').SetInt( 411);
11707:  'IDS_CAP_VIDEDITERR','LongInt').SetInt( 412);
11708:  'IDS_CAP_READONLYFILE','LongInt').SetInt( 413);
11709:  'IDS_CAP_WRITEERROR','LongInt').SetInt( 414);
11710:  'IDS_CAP_NODISKSPACE','LongInt').SetInt( 415);
11711:  'IDS_CAP_SETFILESIZE','LongInt').SetInt( 416);
11712:  'IDS_CAP_SAVEASPERCENT','LongInt').SetInt( 417);
11713:  'IDS_CAP_DRIVER_ERROR','LongInt').SetInt( 418);
11714:  'IDS_CAP_WAVE_OPEN_ERROR','LongInt').SetInt( 419);
11715:  'IDS_CAP_WAVE_ALLOC_ERROR','LongInt').SetInt( 420);
```

```
11716:  'IDS_CAP_WAVE_PREPARE_ERROR','LongInt').SetInt( 421);
11717:  'IDS_CAP_WAVE_ADD_ERROR','LongInt').SetInt( 422);
11718:  'IDS_CAP_WAVE_SIZE_ERROR','LongInt').SetInt( 423);
11719:  'IDS_CAP_VIDEO_OPEN_ERROR','LongInt').SetInt( 424);
11720:  'IDS_CAP_VIDEO_ALLOC_ERROR','LongInt').SetInt( 425);
11721:  'IDS_CAP_VIDEO_PREPARE_ERROR','LongInt').SetInt( 426);
11722:  'IDS_CAP_VIDEO_ADD_ERROR','LongInt').SetInt( 427);
11723:  'IDS_CAP_VIDEO_SIZE_ERROR','LongInt').SetInt( 428);
11724:  'IDS_CAP_FILE_OPEN_ERROR','LongInt').SetInt( 429);
11725:  'IDS_CAP_FILE_WRITE_ERROR','LongInt').SetInt( 430);
11726:  'IDS_CAP_RECORDING_ERROR','LongInt').SetInt( 431);
11727:  'IDS_CAP_RECORDING_ERROR2','LongInt').SetInt( 432);
11728:  'IDS_CAP_AVI_INIT_ERROR','LongInt').SetInt( 433);
11729:  'IDS_CAP_NO_FRAME_CAP_ERROR','LongInt').SetInt( 434);
11730:  'IDS_CAP_NO_PALETTE_WARN','LongInt').SetInt( 435);
11731:  'IDS_CAP_MCI_CONTROL_ERROR','LongInt').SetInt( 436);
11732:  'IDS_CAP_MCI_CANT_STEP_ERROR','LongInt').SetInt( 437);
11733:  'IDS_CAP_NO_AUDIO_CAP_ERROR','LongInt').SetInt( 438);
11734:  'IDS_CAP_AVI_DRAWDIB_ERROR','LongInt').SetInt( 439);
11735:  'IDS_CAP_COMPRESSOR_ERROR','LongInt').SetInt( 440);
11736:  'IDS_CAP_AUDIO_DROP_ERROR','LongInt').SetInt( 441);
11737:  'IDS_CAP_STAT_LIVE_MODE','LongInt').SetInt( 500);
11738:  'IDS_CAP_STAT_OVERLAY_MODE','LongInt').SetInt( 501);
11739:  'IDS_CAP_STAT_CAP_INIT','LongInt').SetInt( 502);
11740:  'IDS_CAP_STAT_CAP_FINI','LongInt').SetInt( 503);
11741:  'IDS_CAP_STAT_PALETTE_BUILD','LongInt').SetInt( 504);
11742:  'IDS_CAP_STAT_OPTPAL_BUILD','LongInt').SetInt( 505);
11743:  'IDS_CAP_STAT_I_FRAMES','LongInt').SetInt( 506);
11744:  'IDS_CAP_STAT_L_FRAMES','LongInt').SetInt( 507);
11745:  'IDS_CAP_STAT_CAP_L_FRAMES','LongInt').SetInt( 508);
11746:  'IDS_CAP_STAT_CAP_AUDIO','LongInt').SetInt( 509);
11747:  'IDS_CAP_STAT_VIDEOCURRENT','LongInt').SetInt( 510);
11748:  'IDS_CAP_STAT_VIDEOAUDIO','LongInt').SetInt( 511);
11749:  'IDS_CAP_STAT_VIDEOONLY','LongInt').SetInt( 512);
11750:  'IDS_CAP_STAT_FRAMESDROPPED','LongInt').SetInt( 513);
11751:  'AVICAP32','String').SetString( 'AVICAP32.dll
11752: end;
11753:
11754: procedure SIRegister_ALFcnMisc(CL: TPSPascalCompiler);
11755: begin
11756:  Function AlBoolToInt( Value : Boolean) : Integer
11757:  Function ALMediumPos( LTotal, LBorder, LObject : integer) : Integer
11758:  Function AlIsValidEmail( const Value : AnsiString) : boolean
11759:  Function AlLocalDateTimeToGMTDateTime( const aLocalDateTime : TDateTime) : TdateTime
11760:  Function ALInc( var x : integer; Count : integer) : Integer
11761:  function ALCopyStr(const aSourceString: AnsiString; aStart, aLength: Integer): AnsiString
11762:  function ALGetStringFromFile(filename: AnsiString; const ShareMode: Word = fmShareDenyWrite):AnsiString;
11763:  procedure ALSaveStringtoFile(Str: AnsiString; filename: AnsiString);
11764:  Function ALIsInteger(const S: AnsiString): Boolean;
11765:  function ALIsDecimal(const S: AnsiString): boolean;
11766:  Function ALStringToWideString(const S: AnsiString; const aCodePage: Word): WideString;
11767:  function AlWideStringToString(const WS: WideString; const aCodePage: Word): AnsiString;
11768:  function  ALQuotedStr(const S: AnsiString; const Quote: AnsiChar = ''''): AnsiString;
11769:  function  ALDequotedStr(const S: AnsiString; AQuote: AnsiChar): AnsiString;
11770:  function  AlUTF8removeBOM(const S: AnsiString): AnsiString;
11771:  Function ALRandomStr1(const aLength: Longint; const aCharset: Array of Char): AnsiString;
11772:  Function ALRandomStr(const aLength: Longint): AnsiString;
11773:  Function ALRandomStrU1(const aLength: Longint; const aCharset: Array of Char): String;
11774:  Function ALRandomStrU(const aLength: Longint): String;
11775: end;
11776:
11777: procedure SIRegister_ALJSONDoc(CL: TPSPascalCompiler);
11778: begin
11779:  Procedure ALJSONToTStrings(const AJsonStr:AnsiString;aLst:TALStrings; const aNullStr:AnsiString;const
       aTrueStr: AnsiString; const aFalseStr : AnsiString)
11780: end;
11781:
11782: procedure SIRegister_ALWindows(CL: TPSPascalCompiler);
11783: begin
11784:   _ALMEMORYSTATUSEX', 'record dwLength : DWORD; dwMemoryLoad : DWO'
11785:    +'RD; ullTotalPhys : Int64; ullAvailPhys : Int64; ullTotalPageFile : Int64; '
11786:    +'ullAvailPageFile : Int64; ullTotalVirtual : Int64; ullAvailVirtual : Int64'
11787:    +'; ullAvailExtendedVirtual : Int64; end
11788:   TALMemoryStatusEx', '_ALMEMORYSTATUSEX
11789:  Function ALGlobalMemoryStatusEx( var lpBuffer : TALMEMORYSTATUSEX) : BOOL
11790:  Function ALInterlockedExchange64( var Target : LONGlONG; Value : LONGLONG) : LONGLONG
11791:  'INVALID_SET_FILE_POINTER','LongInt').SetInt( DWORD ( - 1 ));
11792:  'QUOTA_LIMITS_HARDWS_MIN_DISABLE','LongWord').SetUInt( $2);
11793:  'QUOTA_LIMITS_HARDWS_MIN_ENABLE','LongWord').SetUInt( $1);
11794:  'QUOTA_LIMITS_HARDWS_MAX_DISABLE','LongWord').SetUInt( $8);
11795:  'QUOTA_LIMITS_HARDWS_MAX_ENABLE','LongWord').SetUInt( $4);
11796: end;
11797:
11798: procedure SIRegister_IPCThrd(CL: TPSPascalCompiler);
11799: begin
11800:   SIRegister_THandledObject(CL);
11801:   SIRegister_TEvent(CL);
11802:   SIRegister_TMutex(CL);
11803:   SIRegister_TSharedMem(CL);
```

```
11804:  'TRACE_BUF_SIZE','LongInt').SetInt( 200 * 1024);
11805:  'TRACE_BUFFER','String').SetString( 'TRACE_BUFFER
11806:  'TRACE_MUTEX','String').SetString( 'TRACE_MUTEX
11807:   //PTraceEntry', '^TTraceEntry // will not work
11808:   SIRegister_TIPCTracer(CL);
11809:  'MAX_CLIENTS','LongInt').SetInt( 6);
11810:  'IPCTIMEOUT','LongInt').SetInt( 2000);
11811:  'IPCBUFFER_NAME','String').SetString( 'BUFFER_NAME
11812:  'BUFFER_MUTEX_NAME','String').SetString( 'BUFFER_MUTEX
11813:  'MONITOR_EVENT_NAME','String').SetString( 'MONITOR_EVENT
11814:  'CLIENT_EVENT_NAME','String').SetString( 'CLIENT_EVENT
11815:  'CONNECT_EVENT_NAME','String').SetString( 'CONNECT_EVENT
11816:  'CLIENT_DIR_NAME','String').SetString( 'CLIENT_DIRECTORY
11817:  'CLIENT_DIR_MUTEX','String').SetString( 'DIRECTORY_MUTEX
11818:   FindClass('TOBJECT'),'EMonitorActive
11819:   FindClass('TOBJECT'),'TIPCThread
11820:   TEventKind', '( evMonitorAttach, evMonitorDetach, evMonitorSigna'
11821:    +'l, evMonitorExit, evClientStart, evClientStop, evClientAttach, evClientDet'
11822:    +'ach, evClientSwitch, evClientSignal, evClientExit )
11823:   TClientFlag', '( cfError, cfMouseMove, cfMouseDown, cfResize, cfAttach )
11824:   TClientFlags', 'set of TClientFlag
11825:   //PEventData', '^TEventData // will not work
11826:   TEventData', 'record X : SmallInt; Y : SmallInt; Flag : TClientF'
11827:    +'lag; Flags : TClientFlags; end
11828:   TConnectEvent', 'Procedure ( Sender : TIPCThread; Connecting : Boolean)
11829:   TDirUpdateEvent', 'Procedure ( Sender : TIPCThread)
11830:   TIPCNotifyEvent', 'Procedure ( Sender : TIPCThread; Data : TEventData)
11831:   //PIPCEventInfo', '^TIPCEventInfo // will not work
11832:   TIPCEventInfo','record FID:Integer;FKind:TEventKind;FData:TEventData;end
11833:   SIRegister_TIPCEvent(CL);
11834:   //PClientDirRecords', '^TClientDirRecords // will not work
11835:   SIRegister_TClientDirectory(CL);
11836:   TIPCState', '( stInActive, stDisconnected, stConnected )
11837:   SIRegister_TIPCThread(CL);
11838:   SIRegister_TIPCMonitor(CL);
11839:   SIRegister_TIPCClient(CL);
11840:  Function IsMonitorRunning( var Hndl : THandle) : Boolean
11841: end;
11842:
11843: (*---------------------------------------------------------------------------*)
11844: procedure SIRegister_ALGSMComm(CL: TPSPascalCompiler);
11845: begin
11846:   SIRegister_TAlGSMComm(CL);
11847:  Function AlGSMComm_BuildPDUMessage( aSMSCenter, aSMSAddress, aMessage : AnsiString) : AnsiString
11848:  Procedure AlGSMComm_DecodePDUMessage(aPDUMessage:AnsiString;var aSMSCenter,aSMSAddress,
      AMessage:AnsiString);
11849:  Function AlGSMComm_UnicodeToGSM7BitDefaultAlphabet( aMessage : WideString) : AnsiString
11850:  Function AlGSMComm_GSM7BitDefaultAlphabetToUnicode(aMess:AnsiString;const
      UseGreekAlphabet:Bool):Widestring;
11851:  function ALMatchesMask(const Filename, Mask: AnsiString): Boolean;
11852:  end;
11853:
11854: procedure SIRegister_ALHttpCommon(CL: TPSPascalCompiler);
11855: begin
11856:   TALHTTPPropertyChangeEvent','Procedure(sender:Tobject;const PropertyIndex:Integer;
11857:   TALHTTPProtocolVersion', '( HTTPpv_1_0, HTTPpv_1_1 )
11858:   TALHTTPMethod','(HTTPmt_Get,HTTPmt_Post,HTTPmt_Head,HTTPmt_Trace,HTTPmt_Put,HTTPmt_Delete);
11859:   TInternetScheme', 'integer
11860:   TALIPv6Binary', 'array[1..16] of Char;
11861:  // TALIPv6Binary = array[1..16] of ansiChar;
11862:  //   TInternetScheme = Integer;
11863:   SIRegister_TALHTTPRequestHeader(CL);
11864:   SIRegister_TALHTTPCookie(CL);
11865:   SIRegister_TALHTTPCookieCollection(CL);
11866:   SIRegister_TALHTTPResponseHeader(CL);
11867:  Function ALHTTPDecode( const AStr : AnsiString) : AnsiString
11868:  Procedure ALHTTPEncodeParamNameValues( ParamValues : TALStrings)
11869: // Procedure ALExtractHTTPFields(Separators, WhiteSpace, Quotes:TSysCharSet;
      Content:PAnsiChar;Strings:TALStrings;StripQuotes:Boolean;
11870: // Procedure ALExtractHeaderFields( Separators,WhiteSpace, Quotes : TSysCharSet; Content : PAnsiChar;
      Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11871: // Procedure ALExtractHeaderFieldsWithQuoteEscaped(Separators,WhiteSpace,
      Quotes:TSysCharSet;Content:PAnsiChar;Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11872:  Function AlRemoveShemeFromUrl( aUrl : AnsiString) : ansiString
11873:  Function AlExtractShemeFromUrl( aUrl : AnsiString) : TInternetScheme
11874:  Function AlExtractHostNameFromUrl( aUrl : AnsiString) : AnsiString
11875:  Function AlExtractDomainNameFromUrl( aUrl : AnsiString) : AnsiString
11876:  Function AlExtractUrlPathFromUrl( aUrl : AnsiString) : AnsiString
11877:  Function AlInternetCrackUrl( aUrl : AnsiString; var SchemeName,HostName,UserName,Password,UrlPath,
      ExtraInfo : AnsiString; var PortNumber : integer) : Boolean;
11878:  Function AlInternetCrackUrl1( aUrl : AnsiString; var SchemeName, HostName, UserName, Password,UrlPath,
      Anchor : AnsiString; Query : TALStrings; var PortNumber : integer) : Boolean;
11879:  Function AlInternetCrackUrl2(var Url:AnsiString;var Anchor:AnsiString;Query:TALStrings):Boolean;
11880:  Function AlRemoveAnchorFromUrl( aUrl : AnsiString; var aAnchor : AnsiString) : AnsiString;
11881:  Function AlRemoveAnchorFromUrl1( aUrl : AnsiString) : AnsiString;
11882:  Function AlCombineUrl( RelativeUrl, BaseUrl : AnsiString) : AnsiString;
11883:  Function AlCombineUrl1(RelativeUrl, BaseUrl, Anchor : AnsiString; Query:TALStrings) : AnsiString;
11884:  Function ALGmtDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
11885:  Function ALDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
```

```
11886:  Function ALTryRfc822StrToGMTDateTime( const S : AnsiString; out Value : TDateTime) : Boolean
11887:  Function ALRfc822StrToGMTDateTime( const s : AnsiString) : TDateTime
11888:  Function ALTryIPV4StrToNumeric( aIPv4Str : ansiString; var aIPv4Num : Cardinal) : Boolean
11889:  Function ALIPV4StrToNumeric( aIPv4 : ansiString) : Cardinal
11890:  Function ALNumericToIPv4Str( aIPv4 : Cardinal) : ansiString
11891:  Function ALZeroIpV6 : TALIPv6Binary
11892:  Function ALTryIPV6StrToBinary( aIPv6Str : ansiString; var aIPv6Bin : TALIPv6Binary) : Boolean
11893:  Function ALIPV6StrTobinary( aIPv6 : ansiString) : TALIPv6Binary
11894:  Function ALBinaryToIPv6Str( aIPv6 : TALIPv6Binary) : ansiString
11895:  Function ALBinaryStrToIPv6Binary( aIPV6BinaryStr : ansiString) : TALIPv6Binary
11896: end;
11897:
11898: procedure SIRegister_ALFcnHTML(CL: TPSPascalCompiler);
11899: begin
11900:  Procedure ALUTF8ExtractHTMLText(HtmlCont:AnsiString;LstExtractedResourceText:TALStrings;const
       DecodeHTMLText:Boolean);
11901:  Function ALUTF8ExtractHTMLText1(HtmlContent:AnsiString;const DecodeHTMLText:Boolean): AnsiString;
11902:  Function ALXMLCDataElementEncode( Src : AnsiString) : AnsiString
11903:  Function ALXMLTextElementEncode(Src : AnsiString; const useNumericReference : boolean) : AnsiString
11904:  Function ALUTF8XMLTextElementDecode( const Src : AnsiString) : AnsiString
11905:  Function ALUTF8HTMLEncode(const Src:AnsiStr;const EncodeASCIIHtmlEntities:Bool;const
       useNumRef:bool):AnsiString);
11906:  Function ALUTF8HTMLDecode( const Src : AnsiString) : AnsiString
11907:  Function ALJavascriptEncode( const Src : AnsiString; const useNumericReference : boolean) : AnsiString
11908:  Function ALUTF8JavascriptDecode( const Src : AnsiString) : AnsiString
11909:  Procedure ALHideHtmlUnwantedTagForHTMLHandleTagfunct(var HtmlContent:AnsiString; const
       DeleteBodyOfUnwantedTag : Boolean; const ReplaceUnwantedTagCharBy : AnsiChar)
11910:  Procedure ALCompactHtmlTagParams( TagParams : TALStrings)
11911: end;
11912:
11913: procedure SIRegister_ALInternetMessageCommon(CL: TPSPascalCompiler);
11914: begin
11915:   SIRegister_TALEMailHeader(CL);
11916:   SIRegister_TALNewsArticleHeader(CL);
11917:  Function AlParseEmailAddress(FriendlyEmail:AnsiString;var RealName:AString;const
       decodeRealName:Bool):AnsiString;
11918:  Function AlExtractEmailAddress( FriendlyEmail : AnsiString) : AnsiString
11919:  Function ALMakeFriendlyEmailAddress( aRealName, aEmail : AnsiString) : AnsiString
11920:  Function ALEncodeRealName4FriendlyEmailAddress( aRealName : AnsiString) : AnsiString
11921:  Function AlGenerateInternetMessageID : AnsiString;
11922:  Function AlGenerateInternetMessageID1( ahostname : AnsiString) : AnsiString;
11923:  Function ALDecodeQuotedPrintableString( src : AnsiString) : AnsiString
11924:  Function AlDecodeInternetMessageHeaderInUTF8(aHeaderStr:AnsiString;aDefaultCodePage:Integer):AnsiString;
11925: end;
11926:
11927: (*---------------------------------------------------------------------------*)
11928: procedure SIRegister_ALFcnWinSock(CL: TPSPascalCompiler);
11929: begin
11930:  Function ALHostToIP( HostName : AnsiString; var Ip : AnsiString):Boolean
11931:  Function ALIPAddrToName( IPAddr : AnsiString) : AnsiString
11932:  Function ALgetLocalIPs : TALStrings
11933:  Function ALgetLocalHostName : AnsiString
11934: end;
11935:
11936: procedure SIRegister_ALFcnCGI(CL: TPSPascalCompiler);
11937: begin
11938:  Procedure AlCGIInitDefaultServerVariablesFromWebRequest(WebRequest :
       TALWebRequest;ServerVariables:TALStrings);
11939:  Procedure AlCGIInitDefaultServerVariablesFromWebRequest1(WebRequest: TALWebRequest; ServerVariables :
       TALStrings; ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
11940:  Procedure ALCGIInitDefaultServerVariables( ServerVariables : TALStrings);
11941:  Procedure AlCGIInitDefaultServerVariables1(ServerVars:TALStrings;ScriptName,
       ScriptFileName:AnsiString;Url:AnsiString);
11942:  Procedure AlCGIInitServerVariablesFromWebRequest( WebRequest:TALWebRequest; ServerVariables : TALStrings;
       ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
11943:  Procedure AlCGIExec( InterpreterFilename : AnsiString; ServerVariables : TALStrings; RequestContentStream
       : Tstream; ResponseContentStream : Tstream; ResponseHeader : TALHTTPResponseHeader);
11944:  Procedure AlCGIExec1(ScriptName,ScriptFileName, Url, X_REWRITE_URL, InterpreterFilename:AnsiString;
       WebRequest : TALIsapiRequest;
       overloadedCookies:AnsiString;overloadedQueryString:AnsiString;overloadedReferer: AnsiString;'
11945:  +'overloadedRequestContentStream:Tstream;var
       ResponseContentStr:AnsiString;ResponseHeader:TALHTTPResponseHeader;
11946:  Procedure AlCGIExec2(ScriptName,ScriptFileName,Url,X_REWRITE_URL,
       InterpreterFilename:AnsiString;WebRequest: TALIsapiRequest; var ResponseContentString : AnsiString;
       ResponseHeader : TALHTTPResponseHeader);
11947: end;
11948:
11949: procedure SIRegister_ALFcnExecute(CL: TPSPascalCompiler);
11950: begin
11951:   TStartupInfoA', 'TStartupInfo
11952:  'SE_CREATE_TOKEN_NAME','String').SetString( 'SeCreateTokenPrivilege
11953:  SE_ASSIGNPRIMARYTOKEN_NAME','String').SetString( 'SeAssignPrimaryTokenPrivilege
11954:  SE_LOCK_MEMORY_NAME','String').SetString( 'SeLockMemoryPrivilege
11955:  SE_INCREASE_QUOTA_NAME','String').SetString( 'SeIncreaseQuotaPrivilege
11956:  SE_UNSOLICITED_INPUT_NAME','String').SetString( 'SeUnsolicitedInputPrivilege
11957:  SE_MACHINE_ACCOUNT_NAME','String').SetString( 'SeMachineAccountPrivilege
11958:  SE_TCB_NAME','String').SetString( 'SeTcbPrivilege
11959:  SE_SECURITY_NAME','String').SetString( 'SeSecurityPrivilege
11960:  SE_TAKE_OWNERSHIP_NAME','String').SetString( 'SeTakeOwnershipPrivilege
```

```
11961:  SE_LOAD_DRIVER_NAME','String').SetString( 'SeLoadDriverPrivilege
11962:  SE_SYSTEM_PROFILE_NAME','String').SetString( 'SeSystemProfilePrivilege
11963:  SE_SYSTEMTIME_NAME','String').SetString( 'SeSystemtimePrivilege
11964:  SE_PROF_SINGLE_PROCESS_NAME','String').SetString( 'SeProfileSingleProcessPrivilege
11965:  SE_INC_BASE_PRIORITY_NAME','String').SetString( 'SeIncreaseBasePriorityPrivilege
11966:  SE_CREATE_PAGEFILE_NAME','String').SetString( 'SeCreatePagefilePrivilege
11967:  SE_CREATE_PERMANENT_NAME','String').SetString( 'SeCreatePermanentPrivilege
11968:  SE_BACKUP_NAME','String').SetString( 'SeBackupPrivilege
11969:  SE_RESTORE_NAME','String').SetString( 'SeRestorePrivilege
11970:  SE_SHUTDOWN_NAME','String').SetString( 'SeShutdownPrivilege
11971:  SE_DEBUG_NAME','String').SetString( 'SeDebugPrivilege
11972:  SE_AUDIT_NAME','String').SetString( 'SeAuditPrivilege
11973:  SE_SYSTEM_ENVIRONMENT_NAME','String').SetString( 'SeSystemEnvironmentPrivilege
11974:  SE_CHANGE_NOTIFY_NAME','String').SetString( 'SeChangeNotifyPrivilege
11975:  SE_REMOTE_SHUTDOWN_NAME','String').SetString( 'SeRemoteShutdownPrivilege
11976:  SE_UNDOCK_NAME','String').SetString( 'SeUndockPrivilege
11977:  SE_SYNC_AGENT_NAME','String').SetString( 'SeSyncAgentPrivilege
11978:  SE_ENABLE_DELEGATION_NAME','String').SetString( 'SeEnableDelegationPrivilege
11979:  SE_MANAGE_VOLUME_NAME','String').SetString( 'SeManageVolumePrivilege
11980:  Function AlGetEnvironmentString : AnsiString
11981:  Function ALWinExec32(const FileName,CurrentDir,
       Environment:AnsiString;InStream:Tstream;OutStream:TStream):Dword;
11982:  Function ALWinExec321(const FileName:AnsiString; InputStream:Tstream;OutputStream:TStream):Dword;
11983:  Function ALWinExecAndWait32( FileName : AnsiString; Visibility : integer) : DWORD
11984:  Function ALWinExecAndWait32V2( FileName : AnsiString; Visibility : integer) : DWORD
11985:  Function ALNTSetPrivilege( sPrivilege : AnsiString; bEnabled : Boolean) : Boolean
11986: end;
11987:
11988: procedure SIRegister_ALFcnFile(CL: TPSPascalCompiler);
11989: begin
11990:  Function AlEmptyDirectory(Directory:ansiString;SubDirectory:Bool;IgnoreFiles:array of AnsiString; const
       RemoveEmptySubDirectory : Boolean; const FileNameMask : ansiString; const MinFileAge : TdateTime):Boolean;
11991:  Function AlEmptyDirectory1( Directory : ansiString; SubDirectory : Boolean; const
       RemoveEmptySubDirectory:Bool; const FileNameMask : ansiString; const MinFileAge : TdateTime) : Boolean;
11992:  Function AlCopyDirectory( SrcDirectory, DestDirectory : ansiString; SubDirectory : Boolean; const
       FileNameMask : ansiString; const FailIfExists : Boolean) : Boolean
11993:  Function ALGetModuleName : ansistring
11994:  Function ALGetModuleFileNameWithoutExtension : ansistring
11995:  Function ALGetModulePath : ansiString
11996:  Function AlGetFileSize( const AFileName : ansistring) : int64
11997:  Function AlGetFileVersion( const AFileName : ansistring) : ansiString
11998:  Function ALGetFileCreationDateTime( const aFileName : Ansistring) : TDateTime
11999:  Function ALGetFileLastWriteDateTime( const aFileName : Ansistring) : TDateTime
12000:  Function ALGetFileLastAccessDateTime( const aFileName : Ansistring) : TDateTime
12001:  Procedure ALSetFileCreationDateTime( const aFileName : Ansistring; const aCreationDateTime : TDateTime)
12002:  Function ALIsDirectoryEmpty( const directory : ansiString) : boolean
12003:  Function ALFileExists( const Path : ansiString) : boolean
12004:  Function ALDirectoryExists( const Directory : Ansistring) : Boolean
12005:  Function ALCreateDir( const Dir : Ansistring) : Boolean
12006:  Function ALRemoveDir( const Dir : Ansistring) : Boolean
12007:  Function ALDeleteFile( const FileName : Ansistring) : Boolean
12008:  Function ALRenameFile( const OldName, NewName : ansistring) : Boolean
12009: end;
12010:
12011: procedure SIRegister_ALFcnMime(CL: TPSPascalCompiler);
12012: begin
12013:   NativeInt', 'Integer
12014:   NativeUInt', 'Cardinal
12015:  Function ALMimeBase64EncodeString( const S : AnsiString) : AnsiString
12016:  Function ALMimeBase64EncodeStringNoCRLF( const S : AnsiString) : AnsiString
12017:  Function ALMimeBase64DecodeString( const S : AnsiString) : AnsiString
12018:  Function ALMimeBase64EncodedSize( const InputSize : NativeInt) : NativeInt
12019:  Function ALMimeBase64EncodedSizeNoCRLF( const InputSize : NativeInt) : NativeInt
12020:  Function ALMimeBase64DecodedSize( const InputSize : NativeInt) : NativeInt
12021:  Procedure ALMimeBase64Encode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12022:  Procedure ALMimeBase64EncodeNoCRLF( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12023:  Procedure ALMimeBase64EncodeFullLines( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12024:  Function ALMimeBase64Decode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt) : NativeInt;
12025:  Function ALMimeBase64DecodePartial( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
       InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt;'
12026:  + 'var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12027:  Function ALMimeBase64DecodePartialEnd( out OutputBuffer : TByteDynArray; OutputOffset : NativeInt; const
       ByteBuffer : Cardinal; const ByteBufferSpace : Cardinal) : NativeInt;
12028:  Procedure ALMimeBase64Encode(const InputBuf:TByteDynArray;const InputByteCnt:NatInt;out
       OutputBuf:TByteDynArray);
12029:  Procedure ALMimeBase64EncodeNoCRLF(const InputBuffer:TByteDynArray; const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
12030:  Procedure ALMimeBase64EncodeFullLines(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray);
12031:  Function ALMimeBase64Decode1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer:TByteDynArray):NativeInt;
12032:  Function ALMimeBase64DecodePartial1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
       OutputBuffer: TByteDynArray; var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal) : NativeInt;
12033:  Function ALMimeBase64DecodePartialEnd1(out OutputBuffer:TByteDynArray;const ByteBuffer:Cardinal;const
       ByteBufferSpace:Cardinal):NativeInt;
```

```
12034:  Procedure ALMimeBase64EncodeFile( const InputFileName, OutputFileName : TFileName)
12035:  Procedure ALMimeBase64EncodeFileNoCRLF( const InputFileName, OutputFileName : TFileName)
12036:  Procedure ALMimeBase64DecodeFile( const InputFileName, OutputFileName : TFileName)
12037:  Procedure ALMimeBase64EncodeStream( const InputStream : TStream; const OutputStream : TStream)
12038:  Procedure ALMimeBase64EncodeStreamNoCRLF( const InputStream : TStream; const OutputStream : TStream)
12039:  Procedure ALMimeBase64DecodeStream( const InputStream : TStream; const OutputStream : TStream)
12040:  'cALMimeBase64_ENCODED_LINE_BREAK','LongInt').SetInt( 76);
12041:  'cALMimeBase64_DECODED_LINE_BREAK','LongInt').SetInt( cALMimeBase64_ENCODED_LINE_BREAK div 4 * 3);
12042:  'cALMimeBase64_BUFFER_SIZE','LongInt').SetInt( cALMimeBase64_DECODED_LINE_BREAK * 3 * 4 * 4);
12043:  Procedure ALFillMimeContentTypeByExtList( AMIMEList : TALStrings)
12044:  Procedure ALFillExtByMimeContentTypeList( AMIMEList : TALStrings)
12045:  Function ALGetDefaultFileExtFromMimeContentType( aContentType : AnsiString) : AnsiString
12046:  Function ALGetDefaultMIMEContentTypeFromExt( aExt : AnsiString) : AnsiString
12047: end;
12048:
12049: procedure SIRegister_ALXmlDoc(CL: TPSPascalCompiler);
12050: begin
12051:  'cALXMLNodeMaxListSize','LongInt').SetInt( Maxint div 16);
12052:   FindClass('TOBJECT'),'TALXMLNode
12053:   FindClass('TOBJECT'),'TALXMLNodeList
12054:   FindClass('TOBJECT'),'TALXMLDocument
12055:   TAlXMLParseProcessingInstructionEvent','Procedure (Sender:TObject; const Target,Data:AnsiString)
12056:   TAlXMLParseTextEvent', 'Procedure ( Sender : TObject; const str: AnsiString)
12057:   TAlXMLParseStartElementEvent', 'Procedure ( Sender : TObject; co'
12058:    +'nst Name : AnsiString; const Attributes : TALStrings)
12059:   TAlXMLParseEndElementEvent', 'Procedure ( Sender : TObject; const Name : AnsiString)
12060:   TALXmlNodeType', '( ntReserved, ntElement, ntAttribute, ntText, '
12061:    +'ntCData, ntEntityRef, ntEntity, ntProcessingInstr, ntComment, ntDocument, '
12062:    +'ntDocType, ntDocFragment, ntNotation )
12063:   TALXMLDocOption', '( doNodeAutoCreate, doNodeAutoIndent )
12064:   TALXMLDocOptions', 'set of TALXMLDocOption
12065:   TALXMLParseOption', '( poPreserveWhiteSpace, poIgnoreXMLReferences )
12066:   TALXMLParseOptions', 'set of TALXMLParseOption
12067:   TALXMLPrologItem', '( xpVersion, xpEncoding, xpStandalone )
12068:   PALPointerXMLNodeList', '^TALPointerXMLNodeList // will not work
12069:   SIRegister_EALXMLDocError(CL);
12070:   SIRegister_TALXMLNodeList(CL);
12071:   SIRegister_TALXMLNode(CL);
12072:   SIRegister_TALXmlElementNode(CL);
12073:   SIRegister_TALXmlAttributeNode(CL);
12074:   SIRegister_TALXmlTextNode(CL);
12075:   SIRegister_TALXmlDocumentNode(CL);
12076:   SIRegister_TALXmlCommentNode(CL);
12077:   SIRegister_TALXmlProcessingInstrNode(CL);
12078:   SIRegister_TALXmlCDataNode(CL);
12079:   SIRegister_TALXmlEntityRefNode(CL);
12080:   SIRegister_TALXmlEntityNode(CL);
12081:   SIRegister_TALXmlDocTypeNode(CL);
12082:   SIRegister_TALXmlDocFragmentNode(CL);
12083:   SIRegister_TALXmlNotationNode(CL);
12084:   SIRegister_TALXMLDocument(CL);
12085:  cAlXMLUTF8EncodingStr','String').SetString( 'UTF-8
12086:  cALXmlUTF8HeaderStr','String').SetString('<?xmlversion="1.0"encoding="UTF-8"standalone="yes"?>'+#13#10);
12087:  CALNSDelim','String').SetString( ':
12088:  CALXML','String').SetString( 'xml
12089:  CALVersion','String').SetString( 'version
12090:  CALEncoding','String').SetString( 'encoding
12091:  CALStandalone','String').SetString( 'standalone
12092:  CALDefaultNodeIndent','String').SetString( '
12093:  CALXmlDocument','String').SetString( 'DOCUMENT
12094:  Function ALCreateEmptyXMLDocument( const Rootname : AnsiString) : TalXMLDocument
12095:  Procedure ALClearXMLDocument(const rootname:AnsiString;xmldoc:TalXMLDocument;const
       EncodingStr:AnsiString);
12096:  Function ALFindXmlNodeByChildNodeValue(xmlrec:TalxmlNode;const ChildNodeName,
       ChildNodeValue:AnsiString;const Recurse: Boolean) : TalxmlNode
12097:  Function ALFindXmlNodeByNameAndChildNodeValue( xmlrec : TalxmlNode; const NodeName : ansiString; const
       ChildNodeName, ChildNodeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12098:  Function ALFindXmlNodeByAttribute(xmlrec:TalxmlNode;const AttributeName,AttributeValue:AnsiString;const
       Recurse: Boolean):TalxmlNode
12099:  Function ALFindXmlNodeByNameAndAttribute( xmlrec : TalxmlNode; const NodeName : ansiString; const
       AttributeName, AttributeValue : AnsiString; const Recurse : Boolean) : TalxmlNode
12100:  Function ALExtractAttrValue( const AttrName, AttrLine : AnsiString; const Default : AnsiString) :
       AnsiString
12101: end;
12102:
12103: procedure SIRegister_TeCanvas(CL: TPSPascalCompiler);
12104: //based on TEEProc, TeCanvas, TEEngine, TChart
12105: begin
12106:  'TeePiStep','Double').setExtended( Pi / 180.0);
12107:  'TeeDefaultPerspective','LongInt').SetInt( 100);
12108:  'TeeMinAngle','LongInt').SetInt( 270);
12109:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12110:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12111:  'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ));
12112:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12113:  'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12114:  'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12115:  'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ));
12116:  'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
```

```
12117:  'TA_LEFT','LongInt').SetInt( 0);
12118:  'TA_RIGHT','LongInt').SetInt( 2);
12119:  'TA_CENTER','LongInt').SetInt( 6);
12120:  'TA_TOP','LongInt').SetInt( 0);
12121:  'TA_BOTTOM','LongInt').SetInt( 8);
12122:  'teePATCOPY','LongInt').SetInt( 0);
12123:  'NumCirclePoints','LongInt').SetInt( 64);
12124:  'teeDEFAULT_CHARSET','LongInt').SetInt( 1);
12125:  'teeANTIALIASED_QUALITY','LongInt').SetInt( 4);
12126:  'TA_LEFT','LongInt').SetInt( 0);
12127:  'bs_Solid','LongInt').SetInt( 0);
12128:  'teepf24Bit','LongInt').SetInt( 0);
12129:  'teepfDevice','LongInt').SetInt( 1);
12130:  'CM_MOUSELEAVE','LongInt').SetInt( 10000);
12131:  'CM_SYSCOLORCHANGE','LongInt').SetInt( 10001);
12132:  'DC_BRUSH','LongInt').SetInt( 18);
12133:  'DC_PEN','LongInt').SetInt( 19);
12134:   teeCOLORREF', 'LongWord
12135:   TLogBrush', 'record lbStyle : Integer; lbColor : TColor; lbHatch: Integer; end
12136:   //TNotifyEvent', 'Procedure ( Sender : TObject)
12137:   SIRegister_TFilterRegion(CL);
12138:   SIRegister_IFormCreator(CL);
12139:   SIRegister_TTeeFilter(CL);
12140:   //TFilterClass', 'class of TTeeFilter
12141:   SIRegister_TFilterItems(CL);
12142:   SIRegister_TConvolveFilter(CL);
12143:   SIRegister_TBlurFilter(CL);
12144:   SIRegister_TTeePicture(CL);
12145:   TPenEndStyle', '( esRound, esSquare, esFlat )
12146:   SIRegister_TChartPen(CL);
12147:   SIRegister_TChartHiddenPen(CL);
12148:   SIRegister_TDottedGrayPen(CL);
12149:   SIRegister_TDarkGrayPen(CL);
12150:   SIRegister_TWhitePen(CL);
12151:   SIRegister_TChartBrush(CL);
12152:   TTeeView3DScrolled', 'Procedure ( IsHoriz : Boolean)
12153:   TTeeView3DChangedZoom', 'Procedure ( NewZoom : Integer)
12154:   SIRegister_TView3DOptions(CL);
12155:   FindClass('TOBJECT'),'TTeeCanvas
12156:   TTeeTransparency', 'Integer
12157:   SIRegister_TTeeBlend(CL);
12158:   FindClass('TOBJECT'),'TCanvas3D
12159:   SIRegister_TTeeShadow(CL);
12160:   teeTGradientDirection', '( gdTopBottom, gdBottomTop, gdLeftRight, g'
12161:    +'dRightLeft, gdFromCenter, gdFromTopLeft, gdFromBottomLeft, gdRadial,gdDiagonalUp,gdDiagonalDown )
12162:   FindClass('TOBJECT'),'TSubGradient
12163:   SIRegister_TCustomTeeGradient(CL);
12164:   SIRegister_TSubGradient(CL);
12165:   SIRegister_TTeeGradient(CL);
12166:   SIRegister_TTeeFontGradient(CL);
12167:   SIRegister_TTeeFont(CL);
12168:   TCanvasBackMode', '( cbmNone, cbmTransparent, cbmOpaque )
12169:   TCanvasTextAlign', 'Integer
12170:   TTeeCanvasHandle', 'HDC
12171:   SIRegister_TTeeCanvas(CL);
12172:   TPoint3DFloat', 'record X : Double; Y : Double; Z : Double; end
12173:   SIRegister_TFloatXYZ(CL);
12174:   TPoint3D', 'record x : integer; y : integer; z : Integer; end
12175:   TRGB', 'record blue: byte; green: byte; red: byte; end
12176:   {TRGB=packed record
12177:     Blue  : Byte;
12178:     Green : Byte;
12179:     Red   : Byte;
12180:     //$IFDEF CLX  //Alpha : Byte; // Linux  end;}
12181:
12182: TTeeCanvasCalcPoints', 'Function ( x, z : Integer; var P0, P1 : '
12183:    +'TPoint3D; var Color0, Color1 : TColor) : Boolean
12184: TTeeCanvasSurfaceStyle', '( tcsSolid, tcsWire, tcsDot )
12185: TCanvas3DPlane', '( cpX, cpY, cpZ )
12186:   //IInterface', 'IUnknown
12187:   SIRegister_TCanvas3D(CL);
12188:   SIRegister_TTeeCanvas3D(CL);
12189:   TTrianglePoints', 'Array[0..2] of TPoint;
12190:   TFourPoints', 'Array[0..3] of TPoint;
12191:  Function ApplyDark( Color : TColor; HowMuch : Byte) : TColor
12192:  Function ApplyBright( Color : TColor; HowMuch : Byte) : TColor
12193:  Function Point3D( const x, y, z : Integer) : TPoint3D
12194:  Procedure SwapDouble( var a, b : Double)
12195:  Procedure SwapInteger( var a, b : Integer)
12196:  Procedure RectSize( const R : TRect; var RectWidth, RectHeight : Integer)
12197:  Procedure teeRectCenter( const R : TRect; var X, Y : Integer)
12198:  Function RectFromPolygon( const Points : array of TPoint; NumPoints : Integer): TRect
12199:  Function RectFromTriangle( const Points : TTrianglePoints) : TRect
12200:  Function RectangleInRectangle( const Small, Big : TRect) : Boolean
12201:  Procedure ClipCanvas( ACanvas : TCanvas; const Rect : TRect)
12202:  Procedure UnClipCanvas( ACanvas : TCanvas)
12203:  Procedure ClipEllipse( ACanvas : TTeeCanvas; const Rect : TRect)
12204:  Procedure ClipRoundRectangle(ACanvas:TTeeCanvas;const Rect : TRect; RoundSize : Integer)
12205:  Procedure ClipPolygon(ACanvas:TTeeCanvas;const Points:array of TPoint;NumPoints:Integer)
```

```
12206:  'TeeCharForHeight','String').SetString( 'W
12207:  'DarkerColorQuantity','Byte').SetUInt( 128);
12208:  'DarkColorQuantity','Byte').SetUInt( 64);
12209:   TButtonGetColorProc', 'Function  : TColor
12210:   SIRegister_TTeeButton(CL);
12211:   SIRegister_TButtonColor(CL);
12212:   SIRegister_TComboFlat(CL);
12213:  Procedure TeeSetTeePen(FPen:TPen; APen : TChartPen; AColor : TColor; Handle:TTeeCanvasHandle)
12214:  Function TeePoint( const aX, aY : Integer) : TPoint
12215:  Function TEEPointInRect( const Rect : TRect; const P : TPoint) : Boolean;
12216:  Function PointInRect1( const Rect : TRect; x, y : Integer) : Boolean;
12217:  Function TeeRect( Left, Top, Right, Bottom : Integer) : TRect
12218:  Function OrientRectangle( const R : TRect) : TRect
12219:  Procedure TeeSetBitmapSize( Bitmap : TBitmap; Width, Height : Integer)
12220:  Function PolygonBounds( const P : array of TPoint) : TRect
12221:  Function PolygonInPolygon( const A, B : array of TPoint) : Boolean
12222:  Function RGBValue( const Color : TColor) : TRGB
12223:  Function EditColor( AOwner : TComponent; AColor : TColor) : TColor
12224:  Function EditColorDialog( AOwner : TComponent; var AColor : TColor) : Boolean
12225:  Function PointAtDistance( AFrom, ATo : TPoint; ADist : Integer) : TPoint
12226:  Function TeeCull( const P : TFourPoints) : Boolean;
12227:  Function TeeCull1( const P0, P1, P2 : TPoint) : Boolean;
12228:   TSmoothStretchOption', '( ssBestQuality, ssBestPerformance )
12229:  Procedure SmoothStretch( Src, Dst : TBitmap);
12230:  Procedure SmoothStretch1( Src, Dst : TBitmap; Option : TSmoothStretchOption);
12231:  Function TeeDistance( const x, y : Double) : Double
12232:  Function TeeLoadLibrary( const FileName : String) : HInst
12233:  Procedure TeeFreeLibrary( hLibModule : HMODULE)
12234:  Procedure TeeBlendBitmaps( const Percent : Double; ABitmap, BBitmap : TBitmap; BOrigin : TPoint)
12235:  //Procedure TeeCalcLines( var Lines : TRGBArray; Bitmap : TBitmap)
12236:  Procedure TeeShadowSmooth(Bitmap, Back : TBitmap; Left, Top, Width, Height, horz, vert : Integer;
       Smoothness : Double; FullDraw : Boolean; ACanvas : TCanvas3D; Clip : Boolean)
12237:   SIRegister_ICanvasHyperlinks(CL);
12238:   SIRegister_ICanvasToolTips(CL);
12239:  Function Supports( const Instance : IInterface; const IID : TGUID) : Boolean
12240: end;
12241:
12242: procedure SIRegister_ovcmisc(CL: TPSPascalCompiler);
12243: begin
12244:   TOvcHdc', 'Integer
12245:   TOvcHWND', 'Cardinal
12246:   TOvcHdc', 'HDC
12247:   TOvcHWND', 'HWND
12248:  Function LoadBaseBitmap( lpBitmapName : PChar) : HBITMAP
12249:  Function LoadBaseCursor( lpCursorName : PChar) : HCURSOR
12250:  Function ovCompStruct( const S1, S2, Size : Cardinal) : Integer
12251:  Function DefaultEpoch : Integer
12252:  Function DrawButtonFrame(Canvas:TCanvas;const Client:TRect;IsDown,IsFlat:Bool;Style:TButtonStyle):TRect;
12253:  Procedure FixRealPrim( P : PChar; DC : Char)
12254:  Function GetDisplayString( Canvas : TCanvas; const S : string; MinChars, MaxWidth : Integer) : string
12255:  Function GetLeftButton : Byte
12256:  Function GetNextDlgItem( Ctrl : TOvcHWnd) : hWnd
12257:  Procedure GetRGB( Clr : TColor; var IR, IG, IB : Byte)
12258:  Function GetShiftFlags : Byte
12259:  Function ovCreateRotatedFont( F : TFont; Angle : Integer) : hFont
12260:  Function GetTopTextMargin(Font:TFont;BorderStyle:TBorderStyle; Height:Integer;Ctl3D:Boolean): Integer
12261:  Function ovExtractWord( N : Integer; const S : string; WordDelims : TCharSet) : string
12262:  Function ovIsForegroundTask : Boolean
12263:  Function ovTrimLeft( const S : string) : string
12264:  Function ovTrimRight( const S : string) : string
12265:  Function ovQuotedStr( const S : string) : string
12266:  Function ovWordCount( const S : string; const WordDelims : TCharSet) : Integer
12267:  Function ovWordPosition(const N:Integer;const S: string;const WordDelims : TCharSet) : Integer
12268:  Function PtrDiff( const P1, P2 : PChar) : Word
12269:  Procedure PtrInc( var P, Delta : Word)
12270:  Procedure PtrDec( var P, Delta : Word)
12271:  Procedure FixTextBuffer( InBuf, OutBuf : PChar; OutSize : Integer)
12272:  Procedure TransStretchBlt( DstDC : TOvcHdc; DstX, DstY, DstW, DstH : Integer; SrcDC : TOvcHdc; SrcX, SrcY,
       SrcW, SrcH : Integer; MaskDC : TOvcHdc; MaskX, MaskY : Integer)
12273:  Function ovMinI( X, Y : Integer) : Integer
12274:  Function ovMaxI( X, Y : Integer) : Integer
12275:  Function ovMinL( X, Y : LongInt) : LongInt
12276:  Function ovMaxL( X, Y : LongInt) : LongInt
12277:  Function GenerateComponentName( PF : TWinControl; const Root : string) : string
12278:  Function PartialCompare( const S1, S2 : string) : Boolean
12279:  Function PathEllipsis( const S : string; MaxWidth : Integer) : string
12280:  Function ovCreateDisabledBitmap( FOriginal : TBitmap; OutlineColor : TColor) : TBitmap
12281:  Procedure ovCopyParentImage( Control : TControl; Dest : TCanvas)
12282:  Procedure ovDrawTransparentBitmap( Dest : TCanvas; X, Y, W, H : Integer; Rect : TRect; Bitmap : TBitmap;
       TransparentColor : TColor)
12283:  Procedure DrawTransparentBitmapPrim(DC:TOvcHdc;Bitmap:HBitmap;xStart,yStart,Width,Height:Int;Rect:TRect;
       TransparentColor : TColorRef)
12284:  Function ovWidthOf( const R : TRect) : Integer
12285:  Function ovHeightOf( const R : TRect) : Integer
12286:  Procedure ovDebugOutput( const S : string)
12287:  Function GetArrowWidth( Width, Height : Integer) : Integer
12288:  Procedure StripCharSeq( CharSeq : string; var Str : string)
12289:  Procedure StripCharFromEnd( aChr : Char; var Str : string)
12290:  Procedure StripCharFromFront( aChr : Char; var Str : string)
```

```
12291:  Function SystemParametersInfo( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12292:  Function SystemParametersInfoNCM(uiAction,uiParam:UINT;pvParam:TNonClientMetrics;fWinIni:UINT):BOOL;
12293:  Function SystemParametersInfoA( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT) : BOOL
12294:  Function CreateEllipticRgn( p1, p2, p3, p4 : Integer) : HRGN
12295:  Function CreateEllipticRgnIndirect( const p1 : TRect) : HRGN
12296:  Function CreateFontIndirect( const p1 : TLogFont) : HFONT
12297:  Function CreateMetaFile( p1 : PChar) : HDC
12298:  Function DescribePixelFormat(DC: HDC;p2:Int;p3:UINT;var p4:TPixelFormatDescriptor): BOOL
12299:  Function DrawText(hDC:HDC;lpString:PChar;nCount:Integer; var lpRect : TRect;uFormat:UINT):Integer
12300:  Function DrawTextS(hDC:HDC;lpString:string;nCount:Integer; var lpRect:TRect;uFormat:UINT):Integer
12301:  Function SetMapperFlags( DC : HDC; Flag : DWORD) : DWORD
12302:  Function SetGraphicsMode( hdc : HDC; iMode : Integer) : Integer
12303:  Function SetMapMode( DC : HDC; p2 : Integer) : Integer
12304:  Function SetMetaFileBitsEx( Size : UINT; const Data : PChar) : HMETAFILE
12305:  //Function SetPaletteEntries(Palette:HPALETTE;StartIndex,NumEntries:UINT;var PaletteEntries):UINT
12306:  Function SetPixel( DC : HDC; X, Y : Integer; Color : COLORREF) : COLORREF
12307:  Function SetPixelV( DC : HDC; X, Y : Integer; Color : COLORREF) : BOOL
12308:  //Function SetPixelFormat( DC : HDC; PixelFormat : Integer; FormatDef : PPixelFormatDescriptor) : BOOL
12309:  Function SetPolyFillMode( DC : HDC; PolyFillMode : Integer) : Integer
12310:  Function StretchBlt(DestDC:HDC;X,Y,Width,Height:Int;SrcDC:HDC;XSrc,YSrc,SrcWidth,
       SrcHeight:Int;Rop:DWORD):BOOL
12311:  Function SetRectRgn( Rgn : HRgn; X1, Y1, X2, Y2 : Integer) : BOOL
12312:  Function StretchDIBits(DC : HDC; DestX,DestY,DestWidth,DestHeight,SrcX,SrcY,SrcWidth,
       SrcHeight:Int;Bits:int; var BitsInfo : TBitmapInfo; Usage : UINT; Rop : DWORD) : Integer
12313:  Function SetROP2( DC : HDC; p2 : Integer) : Integer
12314:  Function SetStretchBltMode( DC : HDC; StretchMode : Integer) : Integer
12315:  Function SetSystemPaletteUse( DC : HDC; p2 : UINT) : UINT
12316:  Function SetTextCharacterExtra( DC : HDC; CharExtra : Integer) : Integer
12317:  Function SetTextColor( DC : HDC; Color : COLORREF) : COLORREF
12318:  Function SetTextAlign( DC : HDC; Flags : UINT) : UINT
12319:  Function SetTextJustification( DC : HDC; BreakExtra, BreakCount : Integer) : Integer
12320:  Function UpdateColors( DC : HDC) : BOOL
12321:  Function GetViewportExtEx( DC : HDC; var Size : TSize) : BOOL
12322:  Function GetViewportOrgEx( DC : HDC; var Point : TPoint) : BOOL
12323:  Function GetWindowExtEx( DC : HDC; var Size : TSize) : BOOL
12324:  Function GetWindowOrgEx( DC : HDC; var Point : TPoint) : BOOL
12325:  Function IntersectClipRect( DC : HDC; X1, Y1, X2, Y2 : Integer) : Integer
12326:  Function InvertRgn( DC : HDC; p2 : HRGN) : BOOL
12327:  Function MaskBlt( DestDC : HDC; XDest, YDest, Width, Height : Integer; SrcDC : HDC; XScr, YScr : Integer;
       Mask : HBITMAP; xMask, yMask : Integer; Rop : DWORD) : BOOL
12328:  Function PlgBlt(DestDC:HDC;const PtsArray,SrcDC:HDC;XSrc,YSrc,Widt,Heigh:Int;Mask:HBITMAP;xMask,
       yMask:Int):BOOL;
12329:  Function OffsetClipRgn( DC : HDC; XOffset, YOffset : Integer) : Integer
12330:  Function OffsetRgn( RGN : HRGN; XOffset, YOffset : Integer) : Integer
12331:  Function PatBlt( DC : HDC; X, Y, Width, Height : Integer; Rop : DWORD) : BOOL
12332:  Function Pie( DC : HDC; X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Integer) : BOOL
12333:  Function PlayMetaFile( DC : HDC; MF : HMETAFILE) : BOOL
12334:  Function PaintRgn( DC : HDC; RGN : HRGN) : BOOL
12335:  Function PtInRegion( RGN : HRGN; X, Y : Integer) : BOOL
12336:  Function PtVisible( DC : HDC; X, Y : Integer) : BOOL
12337:  Function RectInRegion( RGN : HRGN; const Rect : TRect) : BOOL
12338:  Function RectVisible( DC : HDC; const Rect : TRect) : BOOL
12339:  Function Rectangle( DC : HDC; X1, Y1, X2, Y2 : Integer) : BOOL
12340:  Function RestoreDC( DC : HDC; SavedDC : Integer) : BOOL
12341: end;
12342:
12343: procedure SIRegister_ovcfiler(CL: TPSPascalCompiler);
12344: begin
12345:   SIRegister_TOvcAbstractStore(CL);
12346:   //PExPropInfo', '^TExPropInfo // will not work
12347: //  TExPropInfo', 'record PI : TPropInfo; AObject : TObject; end
12348:   SIRegister_TOvcPropertyList(CL);
12349:   SIRegister_TOvcDataFiler(CL);
12350:  Procedure UpdateStoredList( AForm : TWinControl; AStoredList : TStrings; FromForm : Boolean)
12351:  Procedure UpdateStoredList1( AForm : TCustomForm; AStoredList : TStrings; FromForm : Boolean)
12352:  Function CreateStoredItem( const CompName, PropName : string) : string
12353:  Function ParseStoredItem( const Item : string; var CompName, PropName : string) : Boolean
12354:  //Function GetPropType( PropInfo : PExPropInfo) : PTypeInfo
12355: end;
12356:
12357: procedure SIRegister_ovccoco(CL: TPSPascalCompiler);
12358: begin
12359:  'ovsetsize','LongInt').SetInt( 16);
12360:  'etSyntax','LongInt').SetInt( 0);
12361:  'etSymantic','LongInt').SetInt( 1);
12362:  'chCR','Char').SetString( #13);
12363:  'chLF','Char').SetString( #10);
12364:  'chLineSeparator','').SetString( chCR);
12365:   SIRegister_TCocoError(CL);
12366:   SIRegister_TCommentItem(CL);
12367:   SIRegister_TCommentList(CL);
12368:   SIRegister_TSymbolPosition(CL);
12369: TGenListType', '( glNever, glAlways, glOnError )
12370: TovBitSet', 'set of Integer
12371:   //PStartTable', '^TStartTable // will not work
12372: 'TovCharSet', 'set of AnsiChar
12373: TAfterGenListEvent', 'Procedure ( Sender : TObject; var PrintErrorCount : boolean)
12374: TCommentEvent', 'Procedure ( Sender : TObject; CommentList : TCommentList)
12375: TCustomErrorEvent', 'Function(Sender:TObject;const ErrorCode: longint;const Data:string): string
```

```
12376: TovErrorEvent', 'Procedure ( Sender : TObject; Error : TCocoError)
12377: TovErrorProc', 'Procedure ( ErrorCode : integer; Symbol : TSymbolP'
12378:    +'osition; const Data : string; ErrorType : integer)
12379: TFailureEvent', 'Procedure ( Sender : TObject; NumErrors : integer)
12380: TGetCH', 'Function ( pos : longint) : char
12381: TStatusUpdateProc', 'Procedure ( Sender : TObject; Status: string; LineNum:integer)
12382:   SIRegister_TCocoRScanner(CL);
12383:   SIRegister_TCocoRGrammar(CL);
12384:  '_EF','Char').SetString( #0);
12385:  '_TAB','Char').SetString( #09);
12386:  '_CR','Char').SetString( #13);
12387:  '_LF','Char').SetString( #10);
12388:  '_EL','').SetString( _CR);
12389:  '_EOF','Char').SetString( #26);
12390:  'LineEnds','TCharSet').SetInt(ord(_CR) or ord(_LF) or ord(_EF));
12391:  'minErrDist','LongInt').SetInt( 2);
12392:  Function ovPadL( S : string; ch : char; L : integer) : string
12393: end;
12394:
12395:   TFormatSettings = record
12396:     CurrencyFormat: Byte;
12397:     NegCurrFormat: Byte;
12398:     ThousandSeparator: Char;
12399:     DecimalSeparator: Char;
12400:     CurrencyDecimals: Byte;
12401:     DateSeparator: Char;
12402:     TimeSeparator: Char;
12403:     ListSeparator: Char;
12404:     CurrencyString: string;
12405:     ShortDateFormat: string;
12406:     LongDateFormat: string;
12407:     TimeAMString: string;
12408:     TimePMString: string;
12409:     ShortTimeFormat: string;
12410:     LongTimeFormat: string;
12411:     ShortMonthNames: array[1..12] of string;
12412:     LongMonthNames: array[1..12] of string;
12413:     ShortDayNames: array[1..7] of string;
12414:     LongDayNames: array[1..7] of string;
12415:     TwoDigitYearCenturyWindow: Word;
12416:   end;
12417:
12418: procedure SIRegister_OvcFormatSettings(CL: TPSPascalCompiler);
12419: begin
12420:  Function ovFormatSettings : TFormatSettings
12421: end;
12422:
12423: procedure SIRegister_ovcstr(CL: TPSPascalCompiler);
12424: begin
12425:   TOvcCharSet', 'set of Char
12426:   ovBTable', 'array[0..255] of Byte
12427:   //BTable = array[0..{$IFDEF UNICODE}{$IFDEF
       HUGE_UNICODE_BMTABLE}$FFFF{$ELSE}$FF{$ENDIF}{$ELSE}$FF{$ENDIF}] of Byte;
12428:  Function BinaryBPChar( Dest : PChar; B : Byte) : PChar
12429:  Function BinaryLPChar( Dest : PChar; L : LongInt) : PChar
12430:  Function BinaryWPChar( Dest : PChar; W : Word) : PChar
12431:  Procedure BMMakeTable( MatchString : PChar; var BT : ovBTable)
12432:  Function BMSearch(var Buffer,BufLength:Cardinal;var BT:ovBTable; MatchString:PChar;var Pos:Cardinal):Bool;
12433:  Function BMSearchUC(var Buffer,BufLength:Card; var BT:ovBTable;MatchString:PChar; var Pos: Card):Boolean
12434:  Function CharStrPChar( Dest : PChar; C : Char; Len : Cardinal) : PChar
12435:  Function DetabPChar( Dest : PChar; Src : PChar; TabSize : Byte) : PChar
12436:  Function HexBPChar( Dest : PChar; B : Byte) : PChar
12437:  Function HexLPChar( Dest : PChar; L : LongInt) : PChar
12438:  Function HexPtrPChar( Dest : PChar; P : TObject) : PChar
12439:  Function HexWPChar( Dest : PChar; W : Word) : PChar
12440:  Function LoCaseChar( C : Char) : Char
12441:  Function OctalLPChar( Dest : PChar; L : LongInt) : PChar
12442:  Function StrChDeletePrim( P : PChar; Pos : Cardinal) : PChar
12443:  Function StrChInsertPrim( Dest : PChar; C : Char; Pos: Cardinal) : PChar
12444:  Function StrChPos( P : PChar; C : Char; var Pos : Cardinal) : Boolean
12445:  Procedure StrInsertChars( Dest : PChar; Ch : Char; Pos, Count : Word)
12446:  Function StrStCopy( Dest, S : PChar; Pos, Count : Cardinal) : PChar
12447:  Function StrStDeletePrim( P : PChar; Pos, Count : Cardinal) : PChar
12448:  Function StrStInsert( Dest, S1, S2 : PChar; Pos : Cardinal) : PChar
12449:  Function StrStInsertPrim( Dest, S : PChar; Pos : Cardinal) : PChar
12450:  Function StrStPos( P, S : PChar; var Pos : Cardinal) : Boolean
12451:  Function StrToLongPChar( S : PChar; var I : LongInt) : Boolean
12452:  Procedure TrimAllSpacesPChar( P : PChar)
12453:  Function TrimEmbeddedZeros( const S : string) : string
12454:  Procedure TrimEmbeddedZerosPChar( P : PChar)
12455:  Function TrimTrailPrimPChar( S : PChar) : PChar
12456:  Function TrimTrailPChar( Dest, S : PChar) : PChar
12457:  Function TrimTrailingZeros( const S : string) : string
12458:  Procedure TrimTrailingZerosPChar( P : PChar)
12459:  Function UpCaseChar( C : Char) : Char
12460:  Function ovcCharInSet( C : Char; const CharSet : TOvcCharSet) : Boolean
12461:  Function ovc32StringIsCurrentCodePage( const S : WideString) : Boolean;
12462:  //Function ovc32StringIsCurrentCodePage1( const S:PWideChar; CP : Cardinal) : Boolean;
12463: end;
```

```
12464:
12465: procedure SIRegister_AfUtils(CL: TPSPascalCompiler);
12466: begin
12467:   //PRaiseFrame', '^TRaiseFrame // will not work
12468:   TRaiseFrame', 'record NextRaise : PRaiseFrame; ExceptAddr : ___Poin'
12469:    +'ter; ExceptObject : TObject; ExceptionRecord : PExceptionRecord; end
12470:  Procedure SafeCloseHandle( var Handle : THandle)
12471:  Procedure ExchangeInteger( X1, X2 : Integer)
12472:  Procedure FillInteger( const Buffer, Size, Value : Integer)
12473:  Function LongMulDiv( Mult1, Mult2, Div1 : Longint) : Longint
12474:  Function afCompareMem( P1, P2 : TObject; Length : Integer) : Boolean
12475:
12476: FILENAME_ADVAPI32     = 'ADVAPI32.DLL';
12477:     function AbortSystemShutdown; external advapi32 name 'AbortSystemShutdownW';
12478: function AbortSystemShutdown(lpMachineName: PKOLChar): BOOL; stdcall;
12479:     function AccessCheckAndAuditAlarm(SubsystemName: PKOLChar;
12480:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12481:       SecurityDescriptor: PSecurityDescriptor; DesiredAccess: DWORD;
12482:       const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12483:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12484:     function AccessCheckByTypeAndAuditAlarm(SubsystemName: PKOLChar;
12485:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12486:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12487:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12488:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12489:       var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12490:     function AccessCheckByTypeResultListAndAuditAlarm(SubsystemName: PKOLChar;
12491:       HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12492:       SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12493:       AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12494:       ObjectTypeListLength: DWORD; const GenericMapping: TGenericMapping;  ObjectCreation: BOOL;
12495:       var GrantedAccess: DWORD; var AccessStatusList:DWORD; var pfGenerateOnClose: BOOL): BOOL; stdcall;
12496:     function BackupEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12497:     function ClearEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12498:     function CreateProcessAsUser(hToken: THandle; lpApplicationName: PKOLChar;
12499:       lpCommandLine: PKOLChar; lpProcessAttributes: PSecurityAttributes;
12500:       lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL;
12501:       dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PKOLChar;
12502:       const lpStartupInfo: TStartupInfo; var lpProcessInformation: TProcessInformation): BOOL; stdcall;
12503:     function GetCurrentHwProfile(var lpHwProfileInfo: THWProfileInfo): BOOL; stdcall;
12504:     function GetFileSecurity(lpFileName: PKOLChar; RequestedInformation: SECURITY_INFORMATION;
12505:       pSecurityDescriptor: PSecurityDescriptor;nLength:DWORD;var lpnLengthNeeded: DWORD):BOOL; stdcall;
12506:     function GetUserName(lpBuffer: PKOLChar; var nSize: DWORD): BOOL; stdcall;
12507:     function InitiateSystemShutdown(lpMachineName, lpMessage: PKOLChar;
12508:       dwTimeout: DWORD; bForceAppsClosed, bRebootAfterShutdown: BOOL): BOOL; stdcall;
12509:     function LogonUser(lpszUsername, lpszDomain, lpszPassword: PKOLChar;
12510:       dwLogonType, dwLogonProvider: DWORD; var phToken: THandle): BOOL; stdcall;
12511:     function LookupAccountName(lpSystemName, lpAccountName: PKOLChar;
12512:       Sid: PSID; var cbSid: DWORD; ReferencedDomainName: PKOLChar;
12513:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12514:     function LookupAccountSid(lpSystemName: PKOLChar; Sid: PSID;
12515:       Name: PKOLChar; var cbName: DWORD; ReferencedDomainName: PKOLChar;
12516:       var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12517:     function LookupPrivilegeDisplayName(lpSystemName, lpName: PKOLChar;
12518:       lpDisplayName: PKOLChar; var cbDisplayName, lpLanguageId: DWORD): BOOL; stdcall;
12519:     function LookupPrivilegeName(lpSystemName: PKOLChar;
12520:       var lpLuid: TLargeInteger; lpName: PKOLChar; var cbName: DWORD): BOOL; stdcall;
12521:     function LookupPrivilegeValue(lpSystemName, lpName: PKOLChar;
12522:       var lpLuid: TLargeInteger): BOOL; stdcall;
12523:     function ObjectCloseAuditAlarm(SubsystemName: PKOLChar;
12524:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12525:     function ObjectDeleteAuditAlarm(SubsystemName: PKOLChar;
12526:       HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12527:     function ObjectOpenAuditAlarm(SubsystemName: PKOLChar; HandleId: Pointer;
12528:       ObjectTypeName: PKOLChar; ObjectName: PKOLChar; pSecurityDescriptor: PSecurityDescriptor;
12529:       ClientToken: THandle; DesiredAccess, GrantedAccess: DWORD;
12530:       var Privileges: TPrivilegeSet; ObjectCreation, AccessGranted: BOOL;
12531:       var GenerateOnClose: BOOL): BOOL; stdcall;
12532:     function ObjectPrivilegeAuditAlarm(SubsystemName: PKOLChar;
12533:       HandleId: Pointer; ClientToken: THandle; DesiredAccess: DWORD;
12534:       var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12535:     function OpenBackupEventLog(lpUNCServerName, lpFileName: PKOLChar): THandle; stdcall;
12536:     function OpenEventLog(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12537:     function PrivilegedServiceAuditAlarm(SubsystemName, ServiceName: PKOLChar;
12538:       ClientToken: THandle; var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12539:     function ReadEventLog(hEventLog: THandle; dwReadFlags, dwRecordOffset: DWORD;
12540:       lpBuffer: Pointer; nNumberOfBytesToRead: DWORD;
12541:       var pnBytesRead, pnMinNumberOfBytesNeeded: DWORD): BOOL; stdcall;
12542:     function RegConnectRegistry(lpMachineName: PKOLChar; hKey: HKEY;
12543:       var phkResult: HKEY): Longint; stdcall;
12544:     function RegCreateKey(hKey: HKEY; lpSubKey: PKOLChar;
12545:       var phkResult: HKEY): Longint; stdcall;
12546:     function RegCreateKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12547:       Reserved: DWORD; lpClass: PKOLChar; dwOptions: DWORD; samDesired: REGSAM;
12548:       lpSecurityAttributes: PSecurityAttributes; var phkResult: HKEY;
12549:       lpdwDisposition: PDWORD): Longint; stdcall;
12550:     function RegDeleteKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12551:     function RegDeleteValue(hKey: HKEY; lpValueName: PKOLChar): Longint; stdcall;
12552:     function
```

```
12553:       var lpcbName: DWORD; lpReserved: Pointer; lpClass: PKOLChar;
12554:       lpcbClass: PDWORD; lpftLastWriteTime: PFileTime): Longint; stdcall;
12555:     function RegEnumKey(hKey:HKEY; dwIndex:DWORD; lpName:PKOLChar; cbName:DWORD):Longint;stdcall;
12556:     function RegEnumValue(hKey: HKEY; dwIndex: DWORD; lpValueName: PKOLChar;
12557:       var lpcbValueName: DWORD; lpReserved: Pointer; lpType: PDWORD;
12558:       lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12559:     function RegLoadKey(hKey: HKEY; lpSubKey, lpFile: PKOLChar): Longint; stdcall;
12560:     function RegOpenKey(hKey: HKEY; lpSubKey: PKOLChar;var phkResult: HKEY):Longint; stdcall;
12561:     function RegOpenKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12562:       ulOptions: DWORD; samDesired: REGSAM; var phkResult: HKEY): Longint; stdcall;
12563:     function RegQueryInfoKey(hKey: HKEY; lpClass: PKOLChar;
12564:       lpcbClass: PDWORD; lpReserved: Pointer;
12565:       lpcSubKeys, lpcbMaxSubKeyLen, lpcbMaxClassLen, lpcValues,
12566:       lpcbMaxValueNameLen, lpcbMaxValueLen, lpcbSecurityDescriptor: PDWORD;
12567:       lpftLastWriteTime: PFileTime): Longint; stdcall;
12568:     function RegQueryMultipleValues(hKey: HKEY; var ValList;
12569:       NumVals: DWORD; lpValueBuf: PKOLChar; var ldwTotsize: DWORD): Longint; stdcall;
12570:     function RegQueryValue(hKey: HKEY; lpSubKey: PKOLChar;
12571:       lpValue: PKOLChar; var lpcbValue: Longint): Longint; stdcall;
12572:     function RegQueryValueEx(hKey: HKEY; lpValueName: PKOLChar;
12573:       lpReserved: Pointer; lpType: PDWORD; lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12574:     function RegReplaceKey(hKey: HKEY; lpSubKey: PKOLChar;
12575:        lpNewFile: PKOLChar; lpOldFile: PKOLChar): Longint; stdcall;
12576:     function RegRestoreKey(hKey: HKEY; lpFile: PKOLChar; dwFlags: DWORD): Longint; stdcall;
12577:     function RegSaveKey(hKey: HKEY; lpFile: PKOLChar;
12578:       lpSecurityAttributes: PSecurityAttributes): Longint; stdcall;
12579:     function RegSetValue(hKey: HKEY; lpSubKey: PKOLChar;
12580:       dwType: DWORD; lpData: PKOLChar; cbData: DWORD): Longint; stdcall;
12581:     function RegSetValueEx(hKey: HKEY; lpValueName: PKOLChar;
12582:       Reserved: DWORD; dwType: DWORD; lpData: Pointer; cbData: DWORD): Longint; stdcall;
12583:     function RegUnLoadKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12584:     function RegisterEventSource(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12585:     function ReportEvent(hEventLog: THandle; wType, wCategory: Word;
12586:       dwEventID: DWORD; lpUserSid: Pointer; wNumStrings: Word;
12587:       dwDataSize: DWORD; lpStrings, lpRawData: Pointer): BOOL; stdcall;
12588:     function SetFileSecurity(lpFileName: PKOLChar; SecurityInformation: SECURITY_INFORMATION;
12589:       pSecurityDescriptor: PSecurityDescriptor): BOOL; stdcall;
12590:
12591:  Function wAddAtom( lpString : PKOLChar) : ATOM
12592:  Function wBeginUpdateResource( pFileName : PKOLChar; bDeleteExistingResources : BOOL) : THandle
12593:  //Function wCallNamedPipe( lpNamedPipeName : PKOLChar; lpInBuffer : Pointer; nInBufferSize : DWORD;
      lpOutBuffer : Pointer; nOutBufferSize : DWORD; var lpBytesRead : DWORD; nTimeOut : DWORD) : BOOL
12594:  //Function wCommConfigDialog( lpszName : PKOLChar; hWnd : HWND; var lpCC : TCommConfig) : BOOL
12595:  Function wCompareString( Locale : LCID; dwCmpFlags : DWORD; lpString1 : PKOLChar; cchCount1 : Integer;
      lpString2 : PKOLChar; cchCount2 : Integer) : Integer
12596:  Function wCopyFile( lpExistingFileName, lpNewFileName : PKOLChar; bFailIfExists : BOOL) : BOOL
12597:  //Function wCopyFileEx( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
      TFNProgressRoutine; lpData : Pointer; pbCancel : PBool; dwCopyFlags : DWORD) : BOOL
12598:  Function wCreateDirectory( lpPathName : PKOLChar; lpSecurityAttributes : PSecurityAttributes) : BOOL
12599:  Function wCreateDirectoryEx(lpTemplateDirectory,
      lpNewDirectory:PKOLChar;lpSecAttrib:PSecurityAttribts):BOOL;
12600:  Function wCreateEvent(lpEventAttribes:PSecurityAttrib;bManualReset,
      bInitialState:BOOL;lpName:PKOLChar):THandle;
12601:  Function wCreateFile( lpFileName : PKOLChar; dwDesiredAccess, dwShareMode : DWORD; lpSecurityAttributes :
      PSecurityAttributes; dwCreationDisposition, dwFlagsAndAttributes:DWORD;hTemplateFile:THandle):THandle
12602:  Function wCreateFileMapping( hFile : THandle; lpFileMappingAttributes : PSecurityAttributes; flProtect,
      dwMaximumSizeHigh, dwMaximumSizeLow : DWORD; lpName : PKOLChar) : THandle
12603:  Function wCreateHardLink(lpFileName,
      lpExistingFileName:PKOLChar;lpSecurityAttributes:PSecurityAttributes):BOOL
12604:  Function
      CreateMailslot(lpName:PKOLChar;MaxMessSize:DWORD;lReadTimeout:DWORD;lpSecurityAttrib:PSecurityAttributes):THandle);
12605:  Function wCreateNamedPipe( lpName : PKOLChar; dwOpenMode, dwPipeMode, nMaxInstances, nOutBufferSize,
      nInBufferSize, nDefaultTimeOut : DWORD; lpSecurityAttributes : PSecurityAttributes) : THandle
12606:  //Function CreateProcess( lpApplicationName : PKOLChar; lpCommandLine : PKOLChar; lpProcessAttributes,
      lpThreadAttributes : PSecurityAttributes; bInheritHandles : BOOL; dwCreationFlags : DWORD; lpEnvironment :
      Pointer; lpCurrentDirectory:PKOLChar;const lpStartupInfo:TStartupInfo;var
      lpProcessInfo:TProcessInformation): BOOL
12607:  Function wCreateSemaphore(lpSemaphoreAttributes: PSecurityAttributes; lInitialCount, lMaximumCount :
      Longint; lpName : PKOLChar) : THandle
12608:  Function
      wCreateWaitableTimer(lpTimerAttributes:PSecurityAttribs;bManualReset:BOOL;lpTimerName:PKOLChar):THandle);
12609:  Function wDefineDosDevice( dwFlags : DWORD; lpDeviceName, lpTargetPath : PKOLChar) : BOOL
12610:  Function wDeleteFile( lpFileName : PKOLChar) : BOOL
12611:  Function wEndUpdateResource( hUpdate : THandle; fDiscard : BOOL) : BOOL
12612:  //Function
      wEnumCalendarInfo(lpCalInfEnumProc:TFNCalInfEnumProc;Locale:LCID;Calendar:CALID;CalType:CALTYPE):BOOL;
12613:  //Function wEnumDateFormats(lpDateFmtEnumProc: TFNDateFmtEnumProc; Locale : LCID; dwFlags : DWORD) : BOOL
12614: // Function wEnumResourceLanguages(hModule:HMODULE;lpType,
      lpName:PChar;lpEnumFunc:ENUMRESLANGPROC;lParam:Longint:BOOL')
12615:  //Function
      wEnumResourceNames(hModule:HMODULE;lpType:PKOLChar;lpEnumFunc:ENUMRESNAMEPROC;lParam:Longint):BOOL;
12616:  //Function wEnumResourceTypes( hModule:HMODULE; lpEnumFunc:ENUMRESTYPEPROC;lParam:Longint):BOOL;
12617:  //Function wEnumSystemCodePages( lpCodePageEnumProc : TFNCodepageEnumProc; dwFlags : DWORD) : BOOL
12618:  //Function wEnumSystemLocales( lpLocaleEnumProc : TFNLocaleEnumProc; dwFlags : DWORD) : BOOL
12619:  //Function wEnumTimeFormats(lpTimeFmtEnumProc:TFNTimeFmtEnumProc;Locale:LCID;dwFlags:DWORD):BOOL;
12620:  Function wExpandEnvironmentStrings( lpSrc : PKOLChar; lpDst : PKOLChar; nSize : DWORD) : DWORD
12621:  Procedure wFatalAppExit( uAction : UINT; lpMessageText : PKOLChar)
12622:  //Function wFillConsoleOutputCharacter( hConsoleOutput : THandle; cCharacter : KOLChar; nLength : DWORD;
      dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
```

```
12623:  Function wFindAtom( lpString : PKOLChar) : ATOM
12624:  Function
        wFindFirstChangeNotification(lpPathName:PKOLChar;bWatchSubtree:BOOL;dwNotifyFilter:DWORD):THandle;
12625:  Function wFindFirstFile( lpFileName : PKOLChar; var lpFindFileData : TWIN32FindData) : THandle
12626:  //Function wFindFirstFileEx( lpFileName : PKOLChar; fInfoLevelId : TFindexInfoLevels; lpFindFileData :
        Pointer; fSearchOp : TFindexSearchOps; lpSearchFilter : Pointer; dwAdditionalFlags : DWORD) : BOOL
12627:  Function wFindNextFile( hFindFile : THandle; var lpFindFileData : TWIN32FindData) : BOOL
12628:  Function wFindResource( hModule : HMODULE; lpName, lpType : PKOLChar) : HRSRC
12629:  Function wFindResourceEx( hModule : HMODULE; lpType, lpName : PKOLChar; wLanguage : Word) : HRSRC
12630:  Function
        wFoldString(dwMapFlags:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Integer):Integer);
12631:  //Function wFormatMessage( dwFlags : DWORD; lpSource : Pointer; dwMessageId : DWORD; dwLanguageId :
        DWORD; lpBuffer : PKOLChar; nSize : DWORD; Arguments : Pointer) : DWORD
12632:  Function wFreeEnvironmentStrings( EnvBlock : PKOLChar) : BOOL
12633:  Function wGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12634:  Function wGetBinaryType( lpApplicationName : PKOLChar; var lpBinaryType : DWORD) : BOOL
12635:  Function wGetCommandLine : PKOLChar
12636:  //Function wGetCompressedFileSize( lpFileName : PKOLChar; lpFileSizeHigh : PDWORD) : DWORD
12637:  Function wGetComputerName( lpBuffer : PKOLChar; var nSize : DWORD) : BOOL
12638:  Function wGetConsoleTitle( lpConsoleTitle : PKOLChar; nSize : DWORD) : DWORD
12639:  //Function wGetCurrencyFormat( Locale : LCID; dwFlags : DWORD; lpValue : PKOLChar; lpFormat :
        PCurrencyFmt; lpCurrencyStr : PKOLChar; cchCurrency : Integer) : Integer
12640:  Function wGetCurrentDirectory( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12641:  //Function wGetDateFormat( Locale : LCID; dwFlags : DWORD; lpDate : PSystemTime; lpFormat : PKOLChar;
        lpDateStr : PKOLChar; cchDate : Integer) : Integer
12642:  //Function wGetDefaultCommConfig( lpszName:PKOLChar;var lpCC : TCommConfig;var lpdwSize:DWORD):BOOL
12643:  Function wGetDiskFreeSpace( lpRootPathName : PKOLChar; var lpSectorsPerCluster, lpBytesPerSector,
        lpNumberOfFreeClusters, lpTotalNumberOfClusters : DWORD) : BOOL
12644:  //Function wGetDiskFreeSpaceEx( lpDirectoryName : PKOLChar; var lpFreeBytesAvailableToCaller,
        lpTotalNumberOfBytes, lpTotalNumberOfFreeBytes : PLargeInteger) : BOOL
12645:  Function wGetDriveType( lpRootPathName : PKOLChar) : UINT
12646:  Function wGetEnvironmentStrings : PKOLChar
12647:  Function wGetEnvironmentVariable( lpName : PKOLChar; lpBuffer : PKOLChar; nSize : DWORD) : DWORD;
12648:  Function wGetFileAttributes( lpFileName : PKOLChar) : DWORD
12649:  //Function
        wGetFileAttributesEx(lpFileName:PKOLChar;fInfoLevelId:TGetFileExInfoLevs;lpFileInform:Pointer):BOOL;
12650:  Function wGetFullPathName(lpFileName:PKOLChar;nBufferLength:WORD;lpBuffer:PKOLChar;var
        lpFilePart:PKOLChar):DWORD;
12651:  //Function wGetLocaleInfo(Locale:LCID; LCType:LCTYPE;lpLCData:PKOLChar;cchData:Integer): Integer
12652:  Function wGetLogicalDriveStrings( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12653:  Function wGetModuleFileName( hModule : HINST; lpFilename : PKOLChar; nSize : DWORD) : DWORD
12654:  Function wGetModuleHandle( lpModuleName : PKOLChar) : HMODULE
12655:  //Function wGetNamedPipeHandleState( hNamedPipe : THandle; lpState, lpCurInstances, lpMaxCollectionCount,
        lpCollectDataTimeout : PDWORD; lpUserName : PKOLChar; nMaxUserNameSize : DWORD) : BOOL
12656:  //Function wGetNumberFormat( Locale : LCID; dwFlags:DWORD; lpValue:PKOLChar; lpFormat:PNumberFmt;
        lpNumberStr : PKOLChar; cchNumber : Integer) : Integer
12657:  Function wGetPrivateProfileInt(lpAppName,lpKeyName:PKOLChar;nDefault:Integer;lpFileName:PKOLChar):UINT;
12658:  Function
        wGetPrivateProfileSection(lpAppName:PKOLChar;lpRetrStr:PKOLChar;nSize:DWORD;pFileName:PKOLChar):DWORD;
12659:  Function wGetPrivateProfileSectionNames(lpszReturnBuffer:PKOLChar;nSize:DWORD;lpFileName:PKOLChar):DWORD;
12660:  Function wGetPrivateProfileString( lpAppName, lpKeyName, lpDefault : PKOLChar;lpReturnedStr: PKOLChar;
        nSize:DWORD; lpFileName : PKOLChar) : DWORD
12661:  Function wGetProfileInt( lpAppName, lpKeyName : PKOLChar; nDefault : Integer) : UINT
12662:  Function wGetProfileSection( lpAppName : PKOLChar; lpReturnedString : PKOLChar; nSize : DWORD) : DWORD
12663:  Function wGetProfileString(lpAppName,lpKeyName,
        lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD):DWORD;
12664:  Function wGetShortPathName( lpszLongPath:PKOLChar;lpszShortPath: PKOLChar; cchBuffer : DWORD) : DWORD
12665:  //Procedure wGetStartupInfo( var lpStartupInfo : TStartupInfo)
12666: // Function wGetStringTypeEx(Locale:LCID; dwInfoType:DWORD;lpSrcStr:PKOLChar;cchSrc:Integer;var
        lpCharType):BOOL
12667:  Function wGetSystemDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12668:  Function wGetTempFileName( lpPathName, lpPrefixString: PKOLChar;uUnique:UINT;lpTempFileName:PKOLChar):UINT
12669:  Function wGetTempPath( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12670: //Function
        wGetTimeFormat(Loc:LCID;dwFlgs:DWORD;lpTime:PSystemTime;lpFrm:PKOLChar;lpTimeStr:PKOLChar;cTime:Int):Int
12671: //Function wGetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
12672: //Function GetVolumeInformation( lpRootPathName : PKOLChar; lpVolumeNameBuffer : PKOLChar; nVolumeNameSize
        : DWORD; lpVolumeSerialNumber : PDWORD; var lpMaximumComponentLength, lpFileSystemFlags : DWORD;
        lpFileSystemNameBuffer : PKOLChar; nFileSystemNameSize : DWORD) : BOOL
12673:  Function wGetWindowsDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12674:  Function wGlobalAddAtom( lpString : PKOLChar) : ATOM
12675:  Function wGlobalFindAtom( lpString : PKOLChar) : ATOM
12676:  Function wGlobalGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12677:  Function wIsBadStringPtr( lpsz : PKOLChar; ucchMax : UINT) : BOOL
12678:  Function
        wLCMapString(Loc:LCID;dwMapFlgs:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Int):Int;
12679:  Function wLoadLibrary( lpLibFileName : PKOLChar) : HMODULE
12680:  Function wLoadLibraryEx( lpLibFileName : PKOLChar; hFile : THandle; dwFlags : DWORD) : HMODULE
12681:  Function wMoveFile( lpExistingFileName, lpNewFileName : PKOLChar) : BOOL
12682:  Function wMoveFileEx( lpExistingFileName, lpNewFileName : PKOLChar; dwFlags : DWORD) : BOOL
12683:  //Function wMoveFileWithProgress( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
        TFNProgressRoutine; lpData : Pointer; dwFlags : DWORD) : BOOL
12684:  Function wOpenEvent( dwDesiredAccess : DWORD; bInheritHandle : BOOL;lpName:PKOLChar) : THandle
12685:  Function wOpenFileMapping( dwDesiredAccess : DWORD; bInheritHandle:BOOL;lpName: PKOLChar):THandle
12686:  Function wOpenMutex( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar) : THandle
12687:  Function wOpenSemaphore( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar):THandle
12688:  Function wOpenWaitableTimer(dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpTimerName:PKOLChar):THandle
12689:  Procedure wOutputDebugString( lpOutputString : PKOLChar)
12690:  //Function wPeekConsoleInput(hConsoleInput:THandle;var lpBuffer:TInputRecord;nLength:DWORD;var
        lpNumberOfEventsRead:DWORD):BOOL;
```

```
12691:  Function wQueryDosDevice( lpDeviceName : PKOLChar; lpTargetPath : PKOLChar; ucchMax : DWORD) : DWORD
12692:  //Function wQueryRecoveryAgents(p1:PKOLChar;var p2:Pointer;var p3:TRecoveryAgentInformation):DWORD
12693:  //Function wReadConsole( hConsoleInput : THandle; lpBuffer : Pointer; nNumberOfCharsToRead : DWORD; var
        lpNumberOfCharsRead : DWORD; lpReserved : Pointer) : BOOL
12694:  //Function wReadConsoleInput(hConsInp:THandle;var lpBuf:TInpRec;nLength:DWORD;var
        lpNumbOfEventsRead:DWORD):BOOL;
12695:  //Function wReadConsoleOutput( hConsoleOutput : THandle; lpBuffer : Pointer; dwBufferSize, dwBufferCoord
        : TCoord; var lpReadRegion : TSmallRect) : BOOL
12696:  //Function wReadConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
        DWORD; dwReadCoord : TCoord; var lpNumberOfCharsRead : DWORD) : BOOL
12697:  Function wRemoveDirectory( lpPathName : PKOLChar) : BOOL
12698:  //Function wScrollConsoleScreenBuffer( hConsoleOutput : THandle; const lpScrollRectangle : TSmallRect;
        lpClipRectangle : PSmallRect; dwDestinationOrigin : TCoord; var lpFill : TCharInfo) : BOOL
12699:  Function wSearchPath( lpPath,lpFileName,lpExtension:PKOLChar;nBufferLength:DWORD;lpBuffer:PKOLChar;var
        lpFilePart:PKOLChar):DWORD;
12700:  Function wSetComputerName( lpComputerName : PKOLChar) : BOOL
12701:  Function wSetConsoleTitle( lpConsoleTitle : PKOLChar) : BOOL
12702:  Function wSetCurrentDirectory( lpPathName : PKOLChar) : BOOL
12703:  //Function wSetDefaultCommConfig( lpszName : PKOLChar; lpCC : PCommConfig; dwSize : DWORD) : BOOL
12704:  Function wSetEnvironmentVariable( lpName, lpValue : PKOLChar) : BOOL
12705:  Function wSetFileAttributes( lpFileName : PKOLChar; dwFileAttributes : DWORD) : BOOL
12706:  //Function wSetLocaleInfo( Locale : LCID; LCType : LCTYPE; lpLCData : PKOLChar) : BOOL
12707:  Function wSetVolumeLabel( lpRootPathName : PKOLChar; lpVolumeName : PKOLChar) : BOOL
12708:  //Function wUpdateResource(hUpdate:THandle;lpType,
        lpName:PKOLChar;wLanguage:Word;lpData:Ptr;cbData:DWORD):BOOL
12709:  Function wVerLanguageName( wLang : DWORD; szLang : PKOLChar; nSize : DWORD) : DWORD
12710:  Function wWaitNamedPipe( lpNamedPipeName : PKOLChar; nTimeOut : DWORD) : BOOL
12711:  //Function wWriteConsole( hConsoleOutput : THandle; const lpBuffer : Pointer; nNumberOfCharsToWrite :
        DWORD; var lpNumberOfCharsWritten : DWORD; lpReserved : Pointer) : BOOL
12712:  //Function wWriteConsoleInput( hConsoleInput : THandle; const lpBuffer : TInputRecord; nLength : DWORD;
        var lpNumberOfEventsWritten : DWORD) : BOOL
12713:  //Function wWriteConsoleOutput(hConsoleOutput:THandle; lpBuffer:Pointer; dwBufferSize,dwBufferCoord :
        TCoord; var lpWriteRegion : TSmallRect) : BOOL
12714:  //Function wWriteConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
        DWORD; dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12715:  Function wWritePrivateProfileSection( lpAppName, lpString, lpFileName : PKOLChar) : BOOL
12716:  Function wWritePrivateProfileString( lpAppName, lpKeyName, lpString, lpFileName : PKOLChar) : BOOL
12717:  Function wWriteProfileSection( lpAppName, lpString : PKOLChar) : BOOL
12718:  Function wWriteProfileString( lpAppName, lpKeyName, lpString : PKOLChar) : BOOL
12719:  Function wlstrcat( lpString1, lpString2 : PKOLChar) : PKOLChar
12720:  Function wlstrcmp( lpString1, lpString2 : PKOLChar) : Integer
12721:  Function wlstrcmpi( lpString1, lpString2 : PKOLChar) : Integer
12722:  Function wlstrcpy( lpString1, lpString2 : PKOLChar) : PKOLChar
12723:  Function wlstrcpyn( lpString1, lpString2 : PKOLChar; iMaxLength : Integer) : PKOLChar
12724:  Function wlstrlen( lpString : PKOLChar) : Integer
12725:  //Function wMultinetGetConnectionPerformance( lpNetResource : PNetResource; lpNetConnectInfoStruc :
        PNetConnectInfoStruct) : DWORD
12726:  //Function wWNetAddConnection2(var lpNetResource:TNetResource;lpPassword,
        lpUserName:PKOLChar;dwFlags:DWORD):DWORD;
12727:  //Function wWNetAddConnection3( hwndOwner : HWND; var lpNetResource:TNetResource;lpPassword,
        lpUserName:PKOLChar; dwFlags : DWORD) : DWORD
12728:  Function wWNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PKOLChar) : DWORD
12729:  Function wWNetCancelConnection2( lpName : PKOLChar; dwFlags : DWORD; fForce : BOOL) : DWORD
12730:  Function wWNetCancelConnection( lpName : PKOLChar; fForce : BOOL) : DWORD
12731:  //Function wWNetConnectionDialog1( var lpConnDlgStruct : TConnectDlgStruct) : DWORD
12732:  //Function wWNetDisconnectDialog1( var lpConnDlgStruct : TDiscDlgStruct) : DWORD
12733:  //Function wWNetEnumResource(hEnum:THandle;var lpcCount:DWORD;lpBuffer:Ptr;var lpBufferSize:DWORD):DWORD;
12734:  Function wWNetGetConnection(lpLocalName:PKOLChar;lpRemoteName:PKOLChar; var lpnLength:DWORD):DWORD;
12735:  Function wWNetGetLastError( var lpError : DWORD; lpErrorBuf : PKOLChar; nErrorBufSize : DWORD; lpNameBuf
        : PKOLChar; nNameBufSize : DWORD) : DWORD
12736:  //Function wWNetGetNetworkInformation(lpProvider:PKOLChar;var lpNetInfoStruct:TNetInfoStruct):DWORD;
12737:  Function wWNetGetProviderName(dwNetType:DWORD;lpProviderName:PKOLChar;var lpBufferSize:DWORD):DWORD;
12738:  //Function wWNetGetResourceParent(lpNetResource:PNetResource;lpBuffer:Pointer;var cbBuffer:DWORD):DWORD;
12739: //Function wWNetGetUniversalName(lpLocalPath:PKOLChar;dwInfoLevel:DWORD;lpBuffer:Ptr;var
        lpBufferSize:DWORD):DWORD;
12740:  Function wWNetGetUser( lpName : PKOLChar; lpUserName : PKOLChar; var lpnLength : DWORD) : DWORD
12741: // Function wWNetOpenEnum(dwScope,dwType,dwUsage:DWORD;lpNetResource:PNetRes;var lphEnum:THandle):DWORD;
12742: // Function wWNetSetConnection( lpName : PKOLChar; dwProperties : DWORD; pvValues : Pointer) : DWORD
12743:  //Function wWNetUseConnection(hwndOwner:HWND;var
        lpNetResource:TNetResource;lpUserID:PKOLChar;lpPassword:PKOLChar; dwFlags:DWORD;lpAccessName:PKOLChar;var
        lpBufferSize:DWORD;var lpResult:DWORD):DWORD
12744:  Function wGetFileVersionInfo(lptstrFilename:PKOLChar;dwHandle,dwLen:DWORD;lpData:Pointer):BOOL;
12745:  Function wGetFileVersionInfoSize( lptstrFilename : PKOLChar; var lpdwHandle : DWORD) : DWORD
12746:  Function wVerFindFile( uFlags : DWORD; szFileName, szWinDir, szAppDir, szCurDir : PKOLChar; var
        lpuCurDirLen : UINT; szDestDir : PKOLChar; var lpuDestDirLen : UINT) : DWORD
12747:  Function wVerInstallFile( uFlags : DWORD; szSrcFileName, szDestFileName, szSrcDir, szDestDir, szCurDir,
        szTmpFile : PKOLChar; var lpuTmpFileLen : UINT) : DWORD
12748: //Function wVerQueryValue(pBlock:Pointer;lpSubBlock:PKOLChar;var lplpBuffer:Ptr;var puLen:UINT):BOOL;
12749: //Function wGetPrivateProfileStruct(lpszSection,
        lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12750: //Function wWritePrivateProfileStruct(lpszSection,
        lpszKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12751:  Function wAddFontResource( FileName : PKOLChar) : Integer
12752:  //Function wAddFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : Integer
12753:  Function wCopyEnhMetaFile( p1 : HENHMETAFILE; p2 : PKOLChar) : HENHMETAFILE
12754:  Function wCopyMetaFile( p1 : HMETAFILE; p2 : PKOLChar) : HMETAFILE
12755:  //Function wCreateColorSpace( var ColorSpace : TLogColorSpace) : HCOLORSPACE
12756:  //Function wCreateDC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12757: // Function wCreateEnhMetaFile( DC : HDC; FileName : PKOLChar; Rect : PRect; Desc : PKOLChar) : HDC
```

```
12758:  Function wCreateFont( nHeight, nWidth, nEscapement, nOrientaion, fnWeight : Integer; fdwItalic,
        fdwUnderline, fdwStrikeOut,fdwCharSet,fdwOutputPrec,fdwClipPrecision,fdwQualy,
        fdwPitchAndFamily:DWORD;lpszFace:PKOLChar):HFONT;
12759:  Function wCreateFontIndirect( const p1 : TLogFont) : HFONT
12760:  //Function wCreateFontIndirectEx( const p1 : PEnumLogFontExDV) : HFONT
12761: // Function wCreateIC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvmInit : PDeviceMode) : HDC
12762:  Function wCreateMetaFile( p1 : PKOLChar) : HDC
12763:  Function wCreateScalableFontResource( p1 : DWORD; p2, p3, p4 : PKOLChar) : BOOL
12764: //Function wDeviceCapabilities(pDriverNa,pDeviceNam,
        pPort:PKOLChar;iIdx:Int;pOut:PKOLChar;DevMod:PDeviceMode):Int;
12765: // Function wEnumFontFamilies( DC : HDC; p2 : PKOLChar; p3 : TFNFontEnumProc; p4 : LPARAM) : BOOL
12766:  //Function wEnumFontFamiliesEx(DC:HDC;var p2:TLogFont;p3:TFNFontEnumProc;p4:LPARAM;p5:DWORD):BOOL);
12767:  //Function wEnumFonts(DC:HDC;lpszFace:PKOLChar;fntenmprc:TFNFontEnumProc;lpszData:PKOLChar):Integer;
12768:  //Function wEnumICMProfiles( DC : HDC; ICMProc : TFNICMEnumProc; p3 : LPARAM) : Integer
12769:  //Function wExtTextOut(DC:HDC;X,
        Y:Int;Options:Longint;Rect:PRect;Str:PKOLChar;Count:Longint;Dx:PInteger:BOOL
12770:  //Function wGetCharABCWidths( DC : HDC; FirstChar, LastChar : UINT; const ABCStructs) : BOOL
12771:  //Function wGetCharABCWidthsFloat( DC : HDC; FirstChar, LastChar : UINT; const ABCFloatSturcts) : BOOL
12772:  //Function wGetCharWidth32( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12773:  //Function wGetCharWidth( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12774: // Function wGetCharWidthFloat( DC : HDC; FirstChar, LastChar : UINT; const Widths) : BOOL
12775: // Function wGetCharacterPlacement(DC:HDC;p2:PKOLChar;p3,p4:BOOL;var p5:TGCPResults;p6:DWORD):DWORD
12776:  Function wGetEnhMetaFile( p1 : PKOLChar) : HENHMETAFILE
12777:  Function wGetEnhMetaFileDescription( p1 : HENHMETAFILE; p2 : UINT; p3 : PKOLChar) : UINT
12778: // Function wGetGlyphIndices( DC : HDC; p2 : PKOLChar; p3 : Integer; p4 : PWORD; p5 : DWORD) : DWORD
12779: // Function wGetGlyphOutline( DC : HDC; uChar,uFormat:UINT;const lpgm:TGlyphMetrics; cbBuffer : DWORD;
        lpvBuffer : Pointer; const lpmat2 : TMat2) : DWORD
12780:  Function wGetICMProfile( DC : HDC; var Size : DWORD; Name : PKOLChar) : BOOL
12781: // Function wGetLogColorSpace( p1 : HCOLORSPACE; var ColorSpace : TLogColorSpace; Size : DWORD) : BOOL
12782:  Function wGetMetaFile( p1 : PKOLChar) : HMETAFILE
12783: // Function wGetObject( p1 : HGDIOBJ; p2 : Integer; p3 : Pointer) : Integer
12784:  //Function wGetOutlineTextMetrics( DC : HDC; p2 : UINT; OTMetricStructs : Pointer) : UINT
12785:  //Function wGetTextExtentExPoint(DC:HDC;p2:PKOLChar; p3,p4:Integer;p5,p6:PInteger;var p7:TSize):BOOL
12786:  Function wGetTextExtentPoint32( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12787:  Function wGetTextExtentPoint( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize) : BOOL
12788:  Function wGetTextFace( DC : HDC; Count : Integer; Buffer : PKOLChar) : Integer
12789:  //Function wGetTextMetrics( DC : HDC; var TM : TTextMetric) : BOOL
12790:  Function wPolyTextOut( DC : HDC; const PolyTextArray, Strings : Integer) : BOOL
12791:  Function wRemoveFontResource( FileName : PKOLChar) : BOOL
12792:  //Function wRemoveFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector) : BOOL
12793:  //Function wResetDC( DC : HDC; const InitData : TDeviceMode) : HDC
12794:  Function wSetICMProfile( DC : HDC; Name : PKOLChar) : BOOL
12795:  //Function wStartDoc( DC : HDC; const p2 : TDocInfo) : Integer
12796:  Function wTextOut( DC : HDC; X, Y : Integer; Str : PKOLChar; Count : Integer) : BOOL
12797:  Function wUpdateICMRegKey( p1 : DWORD; p2, p3 : PKOLChar; p4 : UINT) : BOOL
12798:  Function wwglUseFontBitmaps( DC : HDC; p2, p3, p4 : DWORD) : BOOL
12799:  //Function wwglUseFontOutlines(p1:HDC; p2,p3,p4:DWORD;p5,p6:Single;p7:Int;p8:PGlyphMetricsFloat):BOOL
12800:  Function wAppendMenu( hMenu : HMENU; uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12801:  Function wCallMsgFilter( var lpMsg : TMsg; nCode : Integer) : BOOL
12802:  //Function
        wCallWindowProc(lpPrevWndFunc:TFNWndProc;hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT
12803:  //Function wChangeDisplaySettings( var lpDevMode : TDeviceMode; dwFlags : DWORD) : Longint
12804: // Function wChangeDisplaySettingsEx( lpszDeviceName:PKOLChar;var lpDevMode: TDeviceMode; wnd : HWND;
        dwFlags : DWORD; lParam : Pointer) : Longint
12805:  Function wChangeMenu(hMenu:HMENU;cmd:UINT;lpszNewItem:PKOLChar;cmdInsert:UINT;flags:UINT):BOOL;
12806:  Function wCharLower( lpsz : PKOLChar) : PKOLChar
12807:  Function wCharLowerBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12808:  Function wCharNext( lpsz : PKOLChar) : PKOLChar
12809:  //Function wCharNextEx( CodePage : Word; lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12810:  Function wCharPrev( lpszStart : PKOLChar; lpszCurrent : PKOLChar) : PKOLChar
12811: // Function wCharPrevEx( CodePage : Word; lpStart, lpCurrentChar : LPCSTR; dwFlags : DWORD) : LPSTR
12812:  Function wCharToOem( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
12813:  Function wCharToOemBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
12814:  Function wCharUpper( lpsz : PKOLChar) : PKOLChar
12815:  Function wCharUpperBuff( lpsz : PKOLChar; cchLength : DWORD) : DWORD
12816:  Function wCopyAcceleratorTable( hAccelSrc : HACCEL; var lpAccelDst, cAccelEntries : Integer) : Integer
12817:  Function wCreateAcceleratorTable( var Accel, Count : Integer) : HACCEL
12818:  //Function wCreateDesktop(lpszDesktop,
        lpszDevice:PKOLChar;pDevmode:PDeviceMode;dwFlags:DWORD;dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HDESK
12819:  //Function wCreateDialogIndirectParam( hInstance : HINST; const lpTemplate : TDlgTemplate; hWndParent :
        HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
12820:  //Function wCreateDialogParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND;
        lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : HWND
12821:  Function wCreateMDIWindow( lpClassName, lpWindowName:PKOLChar;dwStyle: DWORD; X,Y,nWidth,nHeight:Integer;
        hWndParent : HWND; hInstance : HINST; lParam : LPARAM) : HWND
12822: //Function wCreateWindowEx(dwExStyle:DWORD;lpClassName:PKOLChar;lpWindowName:PKOLChar;dwStyle DWORD;X,Y,
        nWidth, nHeight:Int WndParent:HWND;hMenu:HMENU;hInstance:HINST;lpParam:Pointer):HWND
12823:  //Function wCreateWindowStation(lpwinsta:PKOLChar;dwReserv,
        dwDesiredAccess:DWORD;lpsa:PSecurityAttribs):HWINSTA;
12824:  Function wDefDlgProc( hDlg : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
12825:  Function wDefFrameProc(hWnd,hWndMDIClient:HWND;uMsg:UINT;wParam:WPARAM;lParam:LPARAM):LRESULT;
12826:  Function wDefMDIChildProc( hWnd : HWND; uMsg : UINT; wParam : WPARAM; lParam: LPARAM):LRESULT;
12827:  Function wDefWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM): LRESULT
12828:  //Function wDialogBoxIndirectParam( hInstance : HINST; const lpDialogTemplate : TDlgTemplate; hWndParent
        : HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM) : Integer
12829:  //Function wDialogBoxParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND; lpDialogFunc
        : TFNDlgProc; dwInitParam : LPARAM) : Integer
12830:  Function wDispatchMessage( const lpMsg : TMsg) : Longint
12831:  Function wDlgDirList(hDlg:HWND;lpPathSpec:PKOLChar;nIDListBox,
        nIDStaticPath:Integer;uFileType:UINT):Integer;
```

```
12832:  Function wDlgDirListComboBox(hDlg:HWND;lpPathSpec:PKOLChar;nIDComboBox,
        nIDStaticPath:Int;uFiletype:UINT):Int;
12833:  Function wDlgDirSelectComboBoxEx( hDlg : HWND; lpString:PKOLChar;nCount,nIDComboBox:Integer): BOOL
12834:  Function wDlgDirSelectEx( hDlg : HWND; lpString : PKOLChar; nCount, nIDListBox : Integer) : BOOL
12835: //Function wDrawState(DC:HDC;Brush:HBRUSH;CBFunc:TFNDrawStateProc;lData:LPARA;wDat:WPARA;x,y,cx,
        cy:Int;Flags:UINT):BOOL;
12836:  Function wDrawText(hDC:HDC;lpString:PKOLChar;nCount:Integer;var lpRect:TRect;uFormat:UINT):Integer;
12837:  Function wFindWindow( lpClassName, lpWindowName : PKOLChar) : HWND
12838:  Function wFindWindowEx( Parent, Child : HWND; ClassName, WindowName : PKOLChar) : HWND
12839:  //Function wGetAltTabInfo(hwnd:HWND;iItem:Int;var
        pati:TAltTabInfo;pszItemText:PKOLChar;cchItemText:UINT):BOOL;
12840: // Function wGetClassInfo( hInstance : HINST; lpClassName : PKOLChar; var lpWndClass : TWndClass) : BOOL
12841:  //Function wGetClassInfoEx( Instance : HINST; Classname : PKOLChar; var WndClass : TWndClassEx) : BOOL
12842:  Function wGetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
12843:  Function wGetClassName( hWnd : HWND; lpClassName : PKOLChar; nMaxCount : Integer) : Integer
12844:  Function wGetClipboardFormatName( format : UINT; lpszFormatName:PKOLChar;cchMaxCount:Integer):Integer;
12845:  Function wGetDlgItemText( hDlg: HWND;nIDDlgItem:Integer;lpString:PKOLChar;nMaxCount:Integer):UINT
12846:  Function wGetKeyNameText( lParam : Longint; lpString : PKOLChar; nSize : Integer) : Integer
12847:  Function wGetKeyboardLayoutName( pwszKLID : PKOLChar) : BOOL
12848:  //Function wGetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; var p4 : TMenuItemInfo) : BOOL
12849:  Function wGetMenuString(hMenu:HMENU;uIDItem:UINT;lpString:PKOLChar;nMaxCount:Integer;uFlag:UINT):Integer;
12850:  Function wGetMessage( var lpMsg : TMsg; hWnd : HWND; wMsgFilterMin, wMsgFilterMax : UINT) : BOOL
12851:  Function wGetProp( hWnd : HWND; lpString : PKOLChar) : THandle
12852:  //Function wGetTabbedTextExtent( hDC : HDC; lpString : PKOLChar; nCount, nTabPositions : Integer; var
        lpnTabStopPositions) : DWORD
12853: //Function wGetUserObjectInformation(hObj:THandle;nIndex:Int;pvInfo:Ptr;nLength:DWORD;var
        lpnLengthNeed:DWORD)BOOL;
12854:  Function wGetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
12855:  Function wGetWindowModuleFileName( hwnd : HWND; pszFileName : PKOLChar; cchFileNameMax : UINT) : UINT
12856:  Function wGetWindowText( hWnd : HWND; lpString : PKOLChar; nMaxCount : Integer) : Integer
12857:  Function wGetWindowTextLength( hWnd : HWND) : Integer
12858: //Function wGrayString(hDC:HDC;hBrush:HBRUSH;lpOutFunc:TFNGrayStrProc;lpDat:LPARA;nCnt,X,Y,nWidt,
        nHeigt:Int):BOOL;
12859:  Function wInsertMenu( hMenu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12860:  //Function wInsertMenuItem( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
12861:  Function wIsCharAlpha( ch : KOLChar) : BOOL
12862:  Function wIsCharAlphaNumeric( ch : KOLChar) : BOOL
12863:  Function wIsCharLower( ch : KOLChar) : BOOL
12864:  Function wIsCharUpper( ch : KOLChar) : BOOL
12865:  Function wIsDialogMessage( hDlg : HWND; var lpMsg : TMsg) : BOOL
12866:  Function wLoadAccelerators( hInstance : HINST; lpTableName : PKOLChar) : HACCEL
12867:  Function wLoadBitmap( hInstance : HINST; lpBitmapName : PKOLChar) : HBITMAP
12868:  Function wLoadCursor( hInstance : HINST; lpCursorName : PKOLChar) : HCURSOR
12869:  Function wLoadCursorFromFile( lpFileName : PKOLChar) : HCURSOR
12870:  Function wLoadIcon( hInstance : HINST; lpIconName : PKOLChar) : HICON
12871:  Function wLoadImage(hInst:HINST;ImageName:PKOLChar;ImageType:UINT;X,Y:Integer;Flags:UINT): THandle
12872:  Function wLoadKeyboardLayout( pwszKLID : PKOLChar; Flags : UINT) : HKL
12873:  Function wLoadMenu( hInstance : HINST; lpMenuName : PKOLChar) : HMENU
12874:  //Function wLoadMenuIndirect( lpMenuTemplate : Pointer) : HMENU
12875:  Function wLoadString(hInstance:HINST;uID: UINT; lpBuffer :PKOLChar;nBufferMax:Integer):Integer
12876:  Function wMapVirtualKey( uCode, uMapType : UINT) : UINT
12877:  Function wMapVirtualKeyEx( uCode, uMapType : UINT; dwhkl : HKL) : UINT
12878:  Function wMessageBox( hWnd : HWND; lpText, lpCaption : PKOLChar; uType : UINT) : Integer
12879:  Function wMessageBoxEx( hWnd:HWND; lpText,lpCaption:PKOLChar;uType:UINT;wLanguageId:Word): Integer
12880:  //Function wMessageBoxIndirect( const MsgBoxParams : TMsgBoxParams) : BOOL
12881:  Function wModifyMenu( hMnu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar) : BOOL
12882:  //Function wOemToAnsi( const lpszSrc : LPCSTR; lpszDst : LPSTR) : BOOL
12883:  //7Function wOemToAnsiBuff( lpszSrc : LPCSTR; lpszDst : LPSTR; cchDstLength : DWORD) : BOOL
12884:  //Function wOemToChar( lpszSrc : PKOLChar; lpszDst : PKOLChar) : BOOL
12885:  Function wOemToCharBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD) : BOOL
12886:  Function wOpenDesktop(lpszDesktop:PKOLChar;dwFlags:DWORD;fInherit:BOOL;dwDesiredAccess:DWORD): HDESK
12887:  Function wOpenWindowStation( lpszWinSta : PKOLChar; fInherit : BOOL; dwDesiredAccess : DWORD): HWINSTA
12888:  Function wPeekMessage( var lpMsg : TMsg; hWnd: HWND;wMsgFilterMin,wMsgFilterMax, wRemoveMsg:UINT):BOOL
12889:  Function wPostMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
12890:  Function wPostThreadMessage(idThread:DWORD;Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
12891:  Function wRealGetWindowClass( hwnd : HWND; pszType : PKOLChar; cchType : UINT) : UINT
12892: // Function wRegisterClass( const lpWndClass : TWndClass) : ATOM
12893: // Function wRegisterClassEx( const WndClass : TWndClassEx) : ATOM
12894:  Function wRegisterClipboardFormat( lpszFormat : PKOLChar) : UINT
12895: // Function wRegisterDeviceNotification(hRecipient:THandle;NotificFilter:Pointer;Flags:DWORD):HDEVNOTIFY
12896:  Function wRegisterWindowMessage( lpString : PKOLChar) : UINT
12897:  Function wRemoveProp( hWnd : HWND; lpString : PKOLChar) : THandle
12898:  Function wSendDlgItemMessage(hDlg:HWND;nIDDlgItem:Integer;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
12899:  Function wSendMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
12900:  //Function wSendMessageCallback( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM;
        lpResultCallBack : TFNSendAsyncProc; dwData : DWORD) : BOOL
12901:  Function wSendMessageTimeout(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM; fuFlags,uTimeout:UINT;var
        lpdwResult:DWORD): LRESULT
12902:  Function wSendNotifyMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : BOOL
12903:  Function wSetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
12904:  Function wSetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PKOLChar) : BOOL
12905:  //Function wSetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
12906:  Function wSetProp( hWnd : HWND; lpString : PKOLChar; hData : THandle) : BOOL
12907: // Function wSetUserObjectInformation(hObj:THandle;nIndex:Integer;pvInfo:Pointer;nLength:DWORD):BOOL
12908:  Function wSetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
12909:  Function wSetWindowText( hWnd : HWND; lpString : PKOLChar) : BOOL
12910:  //Function wSetWindowsHook( nFilterType : Integer; pfnFilterProc : TFNHookProc) : HHOOK
12911:  //Function wSetWindowsHookEx(idHook:Integer;lpfn:TFNHookProc;hmod:HINST;dwThreadId:DWORD):HHOOK;
12912: // Function wSystemParametersInfo( uiAction, uiParam : UINT; pvParam : Pointer; fWinIni: UINT):BOOL
```

```
12913:  Function wTabbedTextOut(hDC:HDC;X,Y:Int;lpString:PKOLChar;nCount,nTabPositions:Int;var
       lpnTabStopPositions,nTabOrigin:Int):Longint;
12914:  Function wTranslateAccelerator( hWnd : HWND; hAccTable : HACCEL; var lpMsg : TMsg) : Integer
12915:  Function wUnregisterClass( lpClassName : PKOLChar; hInstance : HINST) : BOOL
12916:  Function wVkKeyScan( ch : KOLChar) : SHORT
12917:  Function wVkKeyScanEx( ch : KOLChar; dwhkl : HKL) : SHORT
12918:  Function wWinHelp( hWndMain : HWND; lpszHelp : PKOLChar; uCommand : UINT; dwData : DWORD) : BOOL
12919:  Function wwsprintf( Output : PKOLChar; Format : PKOLChar) : Integer
12920:  Function wwvsprintf( Output : PKOLChar; Format : PKOLChar; arglist : va_list) : Integer
12921:
12922: //TestDrive!
12923: 'SID_REVISION','LongInt').SetInt(1);'FILENAME_ADVAPI32','String').SetString('ADVAPI32.DLL
12924:  'PROC_CONVERTSIDTOSTRINGSIDA','String').SetString( 'ConvertSidToStringSidA
12925:  Function GetDomainUserSidS(const domainName:String;const userName:String; var foundDomain:String):String;
12926:  Function GetLocalUserSidStr( const UserName : string) : string
12927:  Function getPid4user( const domain : string; const user : string; var pid : dword) : boolean
12928:  Function Impersonate2User( const domain : string; const user : string) : boolean
12929:  Function GetProcessUserBypid( pid : DWORD; var UserName, Domain : AnsiString) : Boolean
12930:  Function KillProcessbyname( const exename : string; var found : integer) : integer
12931:  Function getWinProcessList : TStringList
12932: end;
12933:
12934: procedure SIRegister_AfSafeSync(CL: TPSPascalCompiler);
12935: begin
12936:  'AfMaxSyncSlots','LongInt').SetInt( 64);
12937:  'AfSynchronizeTimeout','LongInt').SetInt( 2000);
12938:   TAfSyncSlotID', 'DWORD
12939:   TAfSyncStatistics','record MessagesCount:Int;TimeoutMessages:Int;DisabledMessages:Int;end;
12940:   TAfSafeSyncEvent', 'Procedure ( ID : TAfSyncSlotID)
12941:   TAfSafeDirectSyncEvent', 'Procedure
12942:  Function AfNewSyncSlot( const AEvent : TAfSafeSyncEvent) : TAfSyncSlotID
12943:  Function AfReleaseSyncSlot( const ID : TAfSyncSlotID) : Boolean
12944:  Function AfEnableSyncSlot( const ID : TAfSyncSlotID; Enable : Boolean) : Boolean
12945:  Function AfValidateSyncSlot( const ID : TAfSyncSlotID) : Boolean
12946:  Function AfSyncEvent( const ID : TAfSyncSlotID; Timeout : DWORD) : Boolean
12947:  Function AfDirectSyncEvent( Event : TAfSafeDirectSyncEvent; Timeout : DWORD) : Boolean
12948:  Function AfIsSyncMethod : Boolean
12949:  Function AfSyncWnd : HWnd
12950:  Function AfSyncStatistics : TAfSyncStatistics
12951:  Procedure AfClearSyncStatistics
12952: end;
12953:
12954: procedure SIRegister_AfComPortCore(CL: TPSPascalCompiler);
12955: begin
12956:  'fBinary','LongWord').SetUInt( $00000001);
12957:  'fParity','LongWord').SetUInt( $00000002);
12958:  'fOutxCtsFlow','LongWord').SetUInt( $00000004);
12959:  'fOutxDsrFlow','LongWord').SetUInt( $00000008);
12960:  'fDtrControl','LongWord').SetUInt( $00000030);
12961:  'fDtrControlDisable','LongWord').SetUInt( $00000000);
12962:  'fDtrControlEnable','LongWord').SetUInt( $00000010);
12963:  'fDtrControlHandshake','LongWord').SetUInt( $00000020);
12964:  'fDsrSensitivity','LongWord').SetUInt( $00000040);
12965:  'fTXContinueOnXoff','LongWord').SetUInt( $00000080);
12966:  'fOutX','LongWord').SetUInt( $00000100);
12967:  'fInX','LongWord').SetUInt( $00000200);
12968:  'fErrorChar','LongWord').SetUInt( $00000400);
12969:  'fNull','LongWord').SetUInt( $00000800);
12970:  'fRtsControl','LongWord').SetUInt( $00003000);
12971:  'fRtsControlDisable','LongWord').SetUInt( $00000000);
12972:  'fRtsControlEnable','LongWord').SetUInt( $00001000);
12973:  'fRtsControlHandshake','LongWord').SetUInt( $00002000);
12974:  'fRtsControlToggle','LongWord').SetUInt( $00003000);
12975:  'fAbortOnError','LongWord').SetUInt( $00004000);
12976:  'fDummy2','LongWord').SetUInt( $FFFF8000);
12977:   TAfCoreEvent', '( ceOutFree, ceLineEvent, ceNeedReadData, ceException )
12978:   CL.FindClass('TOBJECT'),'EAfComPortCoreError
12979:   FindClass('TOBJECT'),'TAfComPortCore
12980:   TAfComPortCoreEvent', 'Procedure ( Sender : TAfComPortCore; Even'
12981:    +'tKind : TAfCoreEvent; Data : DWORD)
12982:   SIRegister_TAfComPortCoreThread(CL);
12983:   SIRegister_TAfComPortEventThread(CL);
12984:   SIRegister_TAfComPortWriteThread(CL);
12985:   SIRegister_TAfComPortCore(CL);
12986:  Function FormatDeviceName( PortNumber : Integer) : string
12987: end;
12988:
12989: procedure SIRegister_ApplicationFileIO(CL: TPSPascalCompiler);
12990: begin
12991:   TAFIOFileStreamEvent', 'Function ( const fileName : String; mode: Word) : TStream
12992:   TAFIOFileStreamExistsEvent', 'Function ( const fileName : String) : Boolean
12993:   SIRegister_TApplicationFileIO(CL);
12994:   TDataFileCapability', '( dfcRead, dfcWrite )
12995:   TDataFileCapabilities', 'set of TDataFileCapability
12996:   SIRegister_TDataFile(CL);
12997:   //TDataFileClass', 'class of TDataFile
12998:  Function ApplicationFileIODefined : Boolean
12999:  Function CreateFileStream(const fileName: String; mode: WordfmShareDenyNone):TStream
13000:  Function FileStreamExists(const fileName: String) : Boolean
```

```
13001:  //Procedure Register
13002: end;
13003:
13004: procedure SIRegister_ALFBXLib(CL: TPSPascalCompiler);
13005: begin
13006:   TALFBXFieldType', '( uftUnKnown, uftNumeric, uftChar, uftVarchar'
13007:    +', uftCstring, uftSmallint, uftInteger, uftQuad, uftFloat, uftDoublePrecisi'
13008:    +'on, uftTimestamp, uftBlob, uftBlobId, uftDate, uftTime, uftInt64, uftArray, uftNull )
13009:   TALFBXScale', 'Integer
13010:   FindClass('TOBJECT'),'EALFBXConvertError
13011:   SIRegister_EALFBXError(CL);
13012:   SIRegister_EALFBXException(CL);
13013:   FindClass('TOBJECT'),'EALFBXGFixError
13014:   FindClass('TOBJECT'),'EALFBXDSQLError
13015:   FindClass('TOBJECT'),'EALFBXDynError
13016:   FindClass('TOBJECT'),'EALFBXGBakError
13017:   FindClass('TOBJECT'),'EALFBXGSecError
13018:   FindClass('TOBJECT'),'EALFBXLicenseError
13019:   FindClass('TOBJECT'),'EALFBXGStatError
13020:   //EALFBXExceptionClass', 'class of EALFBXError
13021:   TALFBXCharacterSet', '( csNONE, csASCII, csBIG_5, csCYRL, csDOS4'
13022:    +'37, csDOS850, csDOS852, csDOS857, csDOS860, csDOS861, csDOS863, csDOS865, '
13023:    +'csEUCJ_0208, csGB_2312, csISO8859_1, csISO8859_2, csKSC_5601, csNEXT, csOC'
13024:    +'TETS, csSJIS_0208, csUNICODE_FSS, csUTF8, csWIN1250, csWIN1251, csWIN1252,'
13025:    +' csWIN1253, csWIN1254, csDOS737, csDOS775, csDOS858, csDOS862, csDOS864, c'
13026:    +'sDOS866, csDOS869, csWIN1255, csWIN1256, csWIN1257, csISO8859_3, csISO8859'
13027:    +'_4, csISO8859_5, csISO8859_6, csISO8859_7, csISO8859_8, csISO8859_9, csISO'
13028:    +'8859_13, csKOI8R, csKOI8U, csWIN1258, csTIS620, csGBK, csCP943C )
13029:   TALFBXTransParam', '( tpConsistency, tpConcurrency, tpShared, tp'
13030:    +'Protected, tpExclusive, tpWait, tpNowait, tpRead, tpWrite, tpLockRead, tpL'
13031:    +'ockWrite, tpVerbTime, tpCommitTime, tpIgnoreLimbo, tpReadCommitted, tpAuto'
13032:    +'Commit, tpRecVersion, tpNoRecVersion, tpRestartRequests, tpNoAutoUndo, tpLockTimeout )
13033:   TALFBXTransParams', 'set of TALFBXTransParam
13034:  Function ALFBXStrToCharacterSet( const CharacterSet : AnsiString) : TALFBXCharacterSet
13035:  Function ALFBXCreateDBParams( Params : AnsiString; Delimiter : Char) : AnsiString
13036:  Function ALFBXCreateBlobParams( Params : AnsiString; Delimiter : Char) : AnsiString
13037:  'cALFBXMaxParamLength','LongInt').SetInt( 125);
13038:   TALFBXParamsFlag', '( pfNotInitialized, pfNotNullable )
13039:   TALFBXParamsFlags', 'set of TALFBXParamsFlag
13040:   //PALFBXSQLVar', '^TALFBXSQLVar // will not work
13041:   //PALFBXSQLDaData', '^TALFBXSQLDaData // will not work
13042:   TALFBXStatementType', '( stSelect, stInsert, stUpdate, stDelete,'
13043:    +' stDDL, stGetSegment, stPutSegment, stExecProcedure, stStartTrans, stCommi'
13044:    +'t, stRollback, stSelectForUpdate, stSetGenerator, stSavePoint )
13045:   SIRegister_TALFBXSQLDA(CL);
13046:   //PALFBXPtrArray', '^TALFBXPtrArray // will not work
13047:   SIRegister_TALFBXPoolStream(CL);
13048:   //PALFBXBlobData', '^TALFBXBlobData // will not work
13049:   TALFBXBlobData', 'record Size : Integer; Buffer : string; end
13050:   //PALFBXArrayDesc', '^TALFBXArrayDesc // will not work
13051:   //TALFBXArrayDesc', 'TISCArrayDesc
13052:   //TALFBXBlobDesc', 'TISCBlobDesc
13053:   //PALFBXArrayInfo', '^TALFBXArrayInfo // will not work
13054:   //TALFBXArrayInfo', 'record index : Integer; size : integer; info: TALFBXArrayDesc; end
13055:   SIRegister_TALFBXSQLResult(CL);
13056:   //TALFBXSQLResultClass', 'class of TALFBXSQLResult
13057:   SIRegister_TALFBXSQLParams(CL);
13058:   //TALFBXSQLParamsClass', 'class of TALFBXSQLParams
13059:   TALFBXDSQLInfoData', 'record InfoCode : byte; InfoLen : Word; St'
13060:    +'atementType : TALFBXStatementType; end
13061:   FindClass('TOBJECT'),'TALFBXLibrary
13062:   //PALFBXStatusVector', '^TALFBXStatusVector // will not work
13063:   TALFBXOnConnectionLost', 'Procedure ( Lib : TALFBXLibrary)
13064:   //TALFBXOnGetDBExceptionClass', 'Procedure ( Number : Integer; out'
13065:    //+' Excep : EALFBXExceptionClass)
13066:   SIRegister_TALFBXLibrary(CL);
13067:  'cAlFBXDateOffset','LongInt').SetInt( 15018);
13068:  'cALFBXTimeCoeff','LongInt').SetInt( 864000000);
13069:  //Procedure ALFBXDecodeTimeStamp( v : PISCTimeStamp; out DateTime : Double);
13070:  //Procedure ALFBXDecodeTimeStamp1( v : PISCTimeStamp; out TimeStamp : TTimeStamp);
13071:  //Function ALFBXDecodeTimeStamp2( v : PISCTimeStamp) : Double;
13072:  Procedure ALFBXDecodeSQLDate( v : Integer; out Year : SmallInt; out Month, Day : Word)
13073:  Procedure ALFBXDecodeSQLTime(v:Cardinal;out Hour,Minute,Second:Word; out Fractions: LongWord)
13074:  //Procedure ALFBXEncodeTimeStamp( const DateTime : TDateTime; v : PISCTimeStamp);
13075:  //Procedure ALFBXEncodeTimeStamp1( const Date : Integer; v : PISCTimeStamp);
13076:  //Procedure ALFBXEncodeTimeStamp2( const Time : Cardinal; v : PISCTimeStamp);
13077:  Function ALFBXEncodeSQLDate( Year : Integer; Month, Day : Integer) : Integer
13078:  Function ALFBXEncodeSQLTime( Hour, Minute, Second : Word; var Fractions : LongWord): Cardinal
13079:   TALFBXParamType', '( prNone, prByte, prShrt, prCard, prStrg, prIgno )
13080:   TALFBXDPBInfo', 'record Name : AnsiString; ParamType : TALFBXParamType; end
13081:  Function ALFBXSQLQuote( const name : AnsiString) : AnsiString
13082:  Function ALFBXSQLUnQuote( const name : AnsiString) : AnsiString
13083: end;
13084:
13085: procedure SIRegister_ALFBXClient(CL: TPSPascalCompiler);
13086: begin
13087:   TALFBXClientSQLParam', 'record Value : AnsiString; IsNull : Boolean; end
13088:   TALFBXClientSQLParams', 'array of TALFBXClientSQLParam
13089:   TALFBXClientSelectDataSQL', 'record SQL : AnsiString; Params : T'
```

```
13090:    +'ALFBXClientSQLParams; RowTag : AnsiString; ViewTag : AnsiString; Skip : in'
13091:    +'teger; First : Integer; CacheThreshold : Integer; end
13092:   TALFBXClientSelectDataSQLs', 'array of TALFBXClientSelectDataSQL
13093:   TALFBXClientUpdateDataSQL', 'record SQL : AnsiString; Params: TALFBXClientSQLParams; end
13094:   TALFBXClientUpdateDataSQLs', 'array of TALFBXClientUpdateDataSQL
13095:   TALFBXClientMonitoringIOStats', 'record page_reads : int64; page'
13096:    +'_writes : int64; page_fetches : int64; page_marks : int64; end
13097:   SIRegister_TALFBXClient(CL);
13098:   SIRegister_TALFBXConnectionStatementPoolBinTreeNode(CL);
13099:   SIRegister_TALFBXConnectionStatementPoolBinTree(CL);
13100:   SIRegister_TALFBXConnectionWithStmtPoolContainer(CL);
13101:   SIRegister_TALFBXConnectionWithoutStmtPoolContainer(CL);
13102:   SIRegister_TALFBXReadTransactionPoolContainer(CL);
13103:   SIRegister_TALFBXReadStatementPoolContainer(CL);
13104:   SIRegister_TALFBXStringKeyPoolBinTreeNode(CL);
13105:   SIRegister_TALFBXConnectionPoolClient(CL);
13106:   SIRegister_TALFBXEventThread(CL);
13107:   Function AlMySqlClientSlashedStr( const Str : AnsiString) : AnsiString
13108: end;
13109:
13110: procedure SIRegister_ovcBidi(CL: TPSPascalCompiler);
13111: begin
13112: _OSVERSIONINFOA = record
13113:     dwOSVersionInfoSize: DWORD;
13114:     dwMajorVersion: DWORD;
13115:     dwMinorVersion: DWORD;
13116:     dwBuildNumber: DWORD;
13117:     dwPlatformId: DWORD;
13118:     szCSDVersion: array[0..127] of AnsiChar; { Maintenance AnsiString for PSS usage }
13119:   end;
13120:  TOSVersionInfoA', '_OSVERSIONINFOA
13121:   TOSVersionInfo', 'TOSVersionInfoA
13122:  'WS_EX_RIGHT','LongWord').SetUInt( $00001000);
13123:  'WS_EX_LEFT','LongWord').SetUInt( $00000000);
13124:  'WS_EX_RTLREADING','LongWord').SetUInt( $00002000);
13125:  'WS_EX_LTRREADING','LongWord').SetUInt( $00000000);
13126:  'WS_EX_LEFTSCROLLBAR','LongWord').SetUInt( $00004000);
13127:  'WS_EX_RIGHTSCROLLBAR','LongWord').SetUInt( $00000000);
13128:  Function SetProcessDefaultLayout( dwDefaultLayout : DWORD) : BOOL
13129:  'LAYOUT_RTL','LongWord').SetUInt( $00000001);
13130:  'LAYOUT_BTT','LongWord').SetUInt( $00000002);
13131:  'LAYOUT_VBH','LongWord').SetUInt( $00000004);
13132:  'LAYOUT_BITMAPORIENTATIONPRESERVED','LongWord').SetUInt( $00000008);
13133:  'NOMIRRORBITMAP','LongWord').SetUInt( DWORD ( $80000000 ));
13134:  Function SetLayout( dc : HDC; dwLayout : DWORD) : DWORD
13135:  Function GetLayout( dc : hdc) : DWORD
13136:  Function IsBidi : Boolean
13137: Function GetCurrentHwProfile( var lpHwProfileInfo : THWProfileInfo) : BOOL
13138:  Function GetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13139:  Function SetPriorityClass( hProcess : THandle; dwPriorityClass: DWORD) : BOOL
13140:  Function GetPriorityClass( hProcess : THandle) : DWORD
13141:  Function OpenClipboard( hWndNewOwner : HWND) : BOOL
13142:  Function CloseClipboard : BOOL
13143:  Function GetClipboardSequenceNumber : DWORD
13144:  Function GetClipboardOwner : HWND
13145:  Function SetClipboardViewer( hWndNewViewer : HWND) : HWND
13146:  Function GetClipboardViewer : HWND
13147:  Function ChangeClipboardChain( hWndRemove, hWndNewNext : HWND) : BOOL
13148:  Function SetClipboardData( uFormat : UINT; hMem : THandle) : THandle
13149:  Function GetClipboardData( uFormat : UINT) : THandle
13150:  Function RegisterClipboardFormat( lpszFormat : PChar) : UINT
13151:  Function CountClipboardFormats : Integer
13152:  Function EnumClipboardFormats( format : UINT) : UINT
13153:  Function GetClipboardFormatName(format:UINT;lpszFormatName:PChar;cchMaxCount:Integer):Integer
13154:  Function EmptyClipboard : BOOL
13155:  Function IsClipboardFormatAvailable( format : UINT) : BOOL
13156:   Function GetPriorityClipboardFormat( var paFormatPriorityList, cFormats : Integer) : Integer
13157:  Function GetOpenClipboardWindow : HWND
13158:  Function EndDialog( hDlg : HWND; nResult : Integer) : BOOL
13159:  Function GetDlgItem( hDlg : HWND; nIDDlgItem : Integer) : HWND
13160:  Function SetDlgItemInt( hDlg : HWND; nIDDlgItem : Integer; uValue : UINT; bSigned: BOOL): BOOL
13161:  Function GetDlgItemInt(hDlg:HWND;nIDDlgItem:Integer;var lpTranslated:BOOL;bSigned: BOOL): UINT
13162:  Function SetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PChar) : BOOL
13163:  Function CheckDlgButton( hDlg : HWND; nIDButton : Integer; uCheck : UINT) : BOOL
13164:  Function CheckRadioButton( hDlg : HWND; nIDFirstButton, nIDLastButton, nIDCheckButton : Integer) : BOOL
13165:  Function IsDlgButtonChecked( hDlg : HWND; nIDButton : Integer) : UINT
13166:  Function SendDlgItemMessage(hDlg:HWND;nIDDlgItem:Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13167: end;
13168:
13169: procedure SIRegister_DXPUtils(CL: TPSPascalCompiler);
13170: begin
13171:  Function glExecuteAndWait(cmdLine:String;visibility:Word;timeout:Cardinal;killAppOnTimeOut:Bool):Int;
13172:  Function GetTemporaryFilesPath : String
13173:  Function GetTemporaryFileName : String
13174:  Function FindFileInPaths( const fileName, paths : String) : String
13175:  Function PathsToString( const paths : TStrings) : String
13176:  Procedure StringToPaths( const pathsString : String; paths : TStrings)
13177:  //Function MacroExpandPath( const aPath : String) : String
13178: end;
```

```
13179:
13180: procedure SIRegister_ALMultiPartBaseParser(CL: TPSPascalCompiler);
13181: begin
13182:   SIRegister_TALMultiPartBaseContent(CL);
13183:   SIRegister_TALMultiPartBaseContents(CL);
13184:   SIRegister_TAlMultiPartBaseStream(CL);
13185:   SIRegister_TALMultipartBaseEncoder(CL);
13186:   SIRegister_TALMultipartBaseDecoder(CL);
13187:  Function ALMultipartExtractBoundaryFromContentType( aContentType : AnsiString) : AnsiString
13188:  Function ALMultipartExtractSubValueFromHeaderLine(aHeaderLine:AnsiString;aName:AnsiString):AnsiString;
13189:  Function ALMultipartSetSubValueInHeaderLine(aHeaderLine:AnsiString;aName,AValue:AnsiString):AnsiString;
13190: end;
13191:
13192: procedure SIRegister_SmallUtils(CL: TPSPascalCompiler);
13193: begin
13194:   TdriveSize', 'record FreeS : Int64; TotalS : Int64; end
13195:   TWinVerRec', 'record WinPlatform : Integer; WinMajorVersion : In'
13196:    +'teger; WinMinorVersion : Integer; WinBuildNumber : Integer; WinCSDVersion: String; end
13197:  Function aAllocPadedMem( Size : Cardinal) : TObject
13198:  Procedure aFreePadedMem( var P : TObject);
13199:  Procedure aFreePadedMem1( var P : PChar);
13200:  Function aCheckPadedMem( P : Pointer) : Byte
13201:  Function aGetPadMemSize( P : Pointer) : Cardinal
13202:  Function aAllocMem( Size : Cardinal) : Pointer
13203:  Function aStrLen( const Str : PChar) : Cardinal
13204:  Function aStrLCopy( Dest : PChar; const Source : PChar; MaxLen : Cardinal) : PChar
13205:  Function aStrECopy( Dest : PChar; const Source : PChar) : PChar
13206:  Function aStrCopy( Dest : PChar; const Source : PChar) : PChar
13207:  Function aStrEnd( const Str : PChar) : PChar
13208:  Function aStrScan( const Str : PChar; aChr : Char) : PChar
13209:  Function aStrMove( Dest : PChar; const Source : PChar; Count : Cardinal) : PChar
13210:  Function aPCharLength( const Str : PChar) : Cardinal
13211:  Function aPCharUpper( Str : PChar) : PChar
13212:  Function aPCharLower( Str : PChar) : PChar
13213:  Function aStrCat( Dest : PChar; const Source : PChar) : PChar
13214:  Function aLastDelimiter( const Delimiters, S : String) : Integer
13215:  Function aCopyTail( const S : String; Len : Integer) : String
13216:  Function aInt2Thos( I : Int64) : String
13217:  Function aUpperCase( const S : String) : String
13218:  Function aLowerCase( const S : string) : String
13219:  Function aCompareText( const S1, S2 : string) : Integer
13220:  Function aSameText( const S1, S2 : string) : Boolean
13221:  Function aInt2Str( Value : Int64) : String
13222:  Function aStr2Int( const Value : String) : Int64
13223:  Function aStr2IntDef( const S : string; Default : Int64) : Int64
13224:  Function aGetFileExt( const FileName : String) : String
13225:  Function aGetFilePath( const FileName : String) : String
13226:  Function aGetFileName( const FileName : String) : String
13227:  Function aChangeExt( const FileName, Extension : String) : String
13228:  Function aAdjustLineBreaks( const S : string) : string
13229:  Function aGetWindowStr( WinHandle : HWND) : String
13230:  Function aDiskSpace( Drive : String) : TdriveSize
13231:  Function aFileExists( FileName : String) : Boolean
13232:  Function aFileSize( FileName : String) : Int64
13233:  Function aDirectoryExists( const Name : string) : Boolean
13234:  Function aSysErrorMessage( ErrorCode : Integer) : string
13235:  Function aShortPathName( const LongName : string) : string
13236:  Function aGetWindowVer : TWinVerRec
13237:  procedure InitDriveSpacePtr;
13238: end;
13239:
13240: procedure SIRegister_MakeApp(CL: TPSPascalCompiler);
13241: begin
13242:  aZero','LongInt').SetInt( 0);
13243:  'makeappDEF','LongInt').SetInt( - 1);
13244:   'CS_VREDRAW','LongInt').SetInt( DWORD ( 1 ));
13245:  'CS_HREDRAW','LongInt').SetInt( DWORD ( 2 ));
13246:  'CS_KEYCVTWINDOW','LongInt').SetInt( 4);
13247:  'CS_DBLCLKS','LongInt').SetInt( 8);
13248:  'CS_OWNDC','LongWord').SetUInt( $20);
13249:  'CS_CLASSDC','LongWord').SetUInt( $40);
13250:  'CS_PARENTDC','LongWord').SetUInt( $80);
13251:  'CS_NOKEYCVT','LongWord').SetUInt( $100);
13252:  'CS_NOCLOSE','LongWord').SetUInt( $200);
13253:  'CS_SAVEBITS','LongWord').SetUInt( $800);
13254:  'CS_BYTEALIGNCLIENT','LongWord').SetUInt( $1000);
13255:  'CS_BYTEALIGNWINDOW','LongWord').SetUInt( $2000);
13256:  'CS_GLOBALCLASS','LongWord').SetUInt( $4000);
13257:  'CS_IME','LongWord').SetUInt( $10000);
13258:  'CS_DROPSHADOW','LongWord').SetUInt( $20000);
13259:   //PPanelFunc', '^TPanelFunc // will not work
13260:   TPanelStyle', '(psEdge, psTabEdge, psBorder, psTabBorder, psTab, psNone )
13261:   TFontLook', '( flBold, flItalic, flUnderLine, flStrikeOut )
13262:   TFontLooks', 'set of TFontLook
13263:  TMessagefunc','function(hWnd,iMsg,wParam,lParam:Integer):Integer)
13264:  Function SetWinClass(const ClassName:String; pMessFunc: Tmessagefunc; wcStyle : Integer): Word
13265:  Function SetWinClassO( const ClassName : String; pMessFunc : TObject; wcStyle : Integer): Word
13266:  Function SetWinClass( const ClassName : String; pMessFunc : TObject; wcStyle : Integer) : Word
13267:  Function MakeForm(Left,Top,Width,Height:Integer;const Caption:String;WinStyle:Integer):Integer
```

```
13268:  Procedure RunMsgLoop( Show : Boolean)
13269:  Function MakeFont(Height,Width:Integer; const FontName:String; Look:TFontLooks; Roman:Boolean): Integer
13270:  Function MakeButton(Left,Top,Width,Height:Integer;pCaption:PChar;hParent,
        ID_Number:Cardinal;hFont:Int):Int;
13271:  Function MakeListBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Integer
13272:  Function MakeComboBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Int
13273:  Function MakePanel(Left,Top,Width,Height,
        hParent:Int;WndFunc:TPanelFunc;ID_Number:Card;Style:TPanelStyle):Int;
13274:  Function MakeSubMenu(const ItemList : String; ID1, ID2 : Cardinal; hMenu : Integer) : Integer
13275:  Function id4menu( a, b : Byte; c : Byte; d : Byte) : Cardinal
13276:  Procedure DoInitMakeApp   //set first to init formclasscontrol!
13277: end;
13278:
13279: procedure SIRegister_ScreenSaver(CL: TPSPascalCompiler);
13280: begin
13281:   TScreenSaverOption', '( ssoAutoAdjustFormProperties, ssoAutoHook'
13282:    +'KeyboardEvents, ssoAutoHookMouseEvents, ssoEnhancedMouseMoveDetection )
13283:   TScreenSaverOptions', 'set of TScreenSaverOption
13284:  'cDefaultScreenSaverOptions','LongInt').Value.ts32:=ord(ssoAutoAdjustFormProperties) or
        ord(ssoAutoHookKeyboardEvents) or ord(ssoEnhancedMouseMoveDetection);
13285:   TScreenSaverPreviewEvent', 'Procedure ( Sender : TObject; previewHwnd: HWND)
13286:   SIRegister_TScreenSaver(CL);
13287:  //Procedure Register
13288:  Procedure SetScreenSaverPassword
13289: end;
13290:
13291: procedure SIRegister_XCollection(CL: TPSPascalCompiler);
13292: begin
13293:   FindClass('TOBJECT'),'TXCollection
13294:   SIRegister_EFilerException(CL);
13295:   SIRegister_TXCollectionItem(CL);
13296:   //TXCollectionItemClass', 'class of TXCollectionItem
13297:   SIRegister_TXCollection(CL);
13298:  Procedure RegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13299:  Procedure DeRegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13300:  Procedure RegisterXCollectionItemClass( aClass : TXCollectionItemClass)
13301:  Procedure UnregisterXCollectionItemClass( aClass : TXCollectionItemClass)
13302:  Function FindXCollectionItemClass( const className : String) : TXCollectionItemClass
13303:  Function GetXCollectionItemClassesList( baseClass : TXCollectionItemClass) : TList
13304: end;
13305:
13306: procedure SIRegister_XOpenGL(CL: TPSPascalCompiler);
13307: begin
13308:   TMapTexCoordMode', '(mtcmUndefined, mtcmNull, mtcmMain, mtcmDual, mtcmSecond,mtcmArbitrary);
13309:  Procedure xglMapTexCoordToNull
13310:  Procedure xglMapTexCoordToMain
13311:  Procedure xglMapTexCoordToSecond
13312:  Procedure xglMapTexCoordToDual
13313:  Procedure xglMapTexCoordToArbitrary( const units : array of Cardinal);
13314:  Procedure xglMapTexCoordToArbitrary1( const bitWiseUnits : Cardinal);
13315:  Procedure xglMapTexCoordToArbitraryAdd( const bitWiseUnits : Cardinal)
13316:  Procedure xglBeginUpdate
13317:  Procedure xglEndUpdate
13318:  Procedure xglPushState
13319:  Procedure xglPopState
13320:  Procedure xglForbidSecondTextureUnit
13321:  Procedure xglAllowSecondTextureUnit
13322:  Function xglGetBitWiseMapping : Cardinal
13323: end;
13324:
13325: procedure SIRegister_VectorLists(CL: TPSPascalCompiler);
13326: begin
13327:   TBaseListOption', '( bloExternalMemory, bloSetCountResetsMemory)
13328:   TBaseListOptions', 'set of TBaseListOption
13329:   SIRegister_TBaseList(CL);
13330:   SIRegister_TBaseVectorList(CL);
13331:   SIRegister_TAffineVectorList(CL);
13332:   SIRegister_TVectorList(CL);
13333:   SIRegister_TTexPointList(CL);
13334:   SIRegister_TXIntegerList(CL);
13335:   //PSingleArrayList', '^TSingleArrayList // will not work
13336:   SIRegister_TSingleList(CL);
13337:   SIRegister_TByteList(CL);
13338:   SIRegister_TQuaternionList(CL);
13339:  Procedure QuickSortLists( startIndex, endIndex : Integer; refList : TSingleList; objList : TList);
13340:  Procedure QuickSortLists1( startIndex, endIndex : Integer; refList : TSingleList; objList : TBaseList);
13341:  Procedure FastQuickSortLists(startIndex,
        endIndex:Integer;refList:TSingleList;objList:TPersistentObjectList);
13342: end;
13343:
13344: procedure SIRegister_MeshUtils(CL: TPSPascalCompiler);
13345: begin
13346:  Procedure ConvertStripToList( const strip : TAffineVectorList; list : TAffineVectorList);
13347:  Procedure ConvertStripToList1( const strip : TIntegerList; list : TIntegerList);
13348:  Procedure ConvertStripToList2(const strip:TAffineVectorList;const
        indices:TIntegerList;list:TAffineVectorList);
13349:  Procedure ConvertIndexedListToList(const data:TAffineVectlist;const
        indices:TIntegerList;list:TAffineVectorList);
13350:  Function BuildVectorCountOptimizedIndices(const vertices:TAffineVectorList; const
        normals:TAffineVectorList; const texCoords : TAffineVectorList) : TIntegerList
```

```
13351:  Procedure RemapReferences( reference : TAffineVectorList; const indices : TIntegerList);
13352:  Procedure RemapReferences1( reference : TIntegerList; const indices : TIntegerList);
13353:  Procedure RemapAndCleanupReferences( reference : TAffineVectorList; indices : TIntegerList)
13354:  Function RemapIndicesToIndicesMap( remapIndices : TIntegerList) : TIntegerList
13355:  Procedure RemapTrianglesIndices( indices, indicesMap : TIntegerList)
13356:  Procedure RemapIndices( indices, indicesMap : TIntegerList)
13357:  Procedure UnifyTrianglesWinding( indices : TIntegerList)
13358:  Procedure InvertTrianglesWinding( indices : TIntegerList)
13359:  Function BuildNormals( reference : TAffineVectorList; indices : TIntegerList) : TAffineVectorList
13360:  Function BuildNonOrientedEdgesList(triangleIndices:TIntegerList; triangleEdges : TIntegerList;
       edgesTriangles : TIntegerList) : TIntegerList
13361:  Procedure WeldVertices( vertices : TAffineVectorList; indicesMap : TIntegerList; weldRadius : Single)
13362:  Function StripifyMesh(indices:TIntegerList;maxVertexIndex:Integer;agglomerateLoneTriangles:Boolean):
       TPersistentObjectList;
13363:  Procedure IncreaseCoherency( indices : TIntegerList; cacheSize : Integer)
13364:  Procedure SubdivideTriangles( smoothFactor : Single; vertices : TAffineVectorList; triangleIndices :
       TIntegerList; normals : TAffineVectorList; onSubdivideEdge : TSubdivideEdgeEvent)
13365: end;
13366:
13367: procedure SIRegister_JclSysUtils(CL: TPSPascalCompiler);
13368: begin
13369:  Procedure GetAndFillMem( var P : TObject; const Size : Integer; const Value : Byte)
13370:  Procedure FreeMemAndNil( var P : TObject)
13371:  Function PCharOrNil( const S : string) : PChar
13372:   SIRegister_TJclReferenceMemoryStream(CL);
13373:   FindClass('TOBJECT'),'EJclVMTError
13374:  {Function GetVirtualMethodCount( AClass : TClass) : Integer
13375:  Function GetVirtualMethod( AClass : TClass; const Index : Integer) : Pointer
13376:  Procedure SetVirtualMethod( AClass : TClass; const Index : Integer; const Method:Pointer)
13377:   PDynamicIndexList', '^TDynamicIndexList // will not work
13378:   PDynamicAddressList', '^TDynamicAddressList // will not work
13379:  Function GetDynamicMethodCount( AClass : TClass) : Integer
13380:  Function GetDynamicIndexList( AClass : TClass) : PDynamicIndexList
13381:  Function GetDynamicAddressList( AClass : TClass) : PDynamicAddressList
13382:  Function HasDynamicMethod( AClass : TClass; Index : Integer) : Boolean
13383:  Function GetDynamicMethod( AClass : TClass; Index : Integer) : Pointer
13384:  Function GetInitTable( AClass : TClass) : PTypeInfo
13385:   PFieldEntry', '^TFieldEntry // will not work}
13386:   TFieldEntry', 'record OffSet : Integer; IDX : Word; Name : Short'
13387:    +'String; end
13388:  Function JIsClass( Address : Pointer) : Boolean
13389:  Function JIsObject( Address : Pointer) : Boolean
13390:  Function GetImplementorOfInterface( const I : IInterface) : TObject
13391:   TDigitCount', 'Integer
13392:   SIRegister_TJclNumericFormat(CL);
13393:  Function JIntToStrZeroPad( Value, Count : Integer) : AnsiString
13394:   TTextHandler', 'Procedure ( const Text : string)
13395: // 'ABORT_EXIT_CODE','LongInt').SetInt( ERROR_CANCELLED 1223);
13396:  Function JExecute(const
       CommandLine:string;OutputLineCallback:TTextHandler;RawOutpt:Bool;AbortPtr:PBool):Card;
13397:  Function JExecute1(const CommandLine:string;var Output:string; RawOutput:Bool; AbortPtr:PBool):Cardinal;
13398:  Function ReadKey : Char   //to and from the DOS console !
13399:   TModuleHandle', 'HINST
13400:   //TModuleHandle', 'Pointer
13401:  'INVALID_MODULEHANDLE_VALUE','LongInt').SetInt( TModuleHandle ( 0 ));
13402:  Function LoadModule( var Module : TModuleHandle; FileName : string) : Boolean
13403:  Function LoadModuleEx( var Module : TModuleHandle; FileName : string; Flags : Cardinal) : Boolean
13404:  Procedure UnloadModule( var Module : TModuleHandle)
13405:  Function GetModuleSymbol( Module : TModuleHandle; SymbolName : string) : Pointer
13406:  Function GetModuleSymbolEx( Module : TModuleHandle; SymbolName : string; var Accu : Boolean) : Pointer
13407:  Function ReadModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size: Cardinal):Boolean;
13408:  Function WriteModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size:Cardinal):Boolean;
13409:   FindClass('TOBJECT'),'EJclConversionError
13410:  Function JStrToBoolean( const S : string) : Boolean
13411:  Function JBooleanToStr( B : Boolean) : string
13412:  Function JIntToBool( I : Integer) : Boolean
13413:  Function JBoolToInt( B : Boolean) : Integer
13414:  'ListSeparator','String').SetString( ';
13415:  'ListSeparator1','String').SetString( ':
13416:  Procedure ListAddItems( var List : string; const Separator, Items : string)
13417:  Procedure ListIncludeItems( var List : string; const Separator, Items : string)
13418:  Procedure ListRemoveItems( var List : string; const Separator, Items : string)
13419:  Procedure ListDelItem( var List : string; const Separator : string; const Index : Integer)
13420:  Function ListItemCount( const List, Separator : string) : Integer
13421:  Function ListGetItem( const List, Separator : string; const Index : Integer) : string
13422:  Procedure ListSetItem(var List:string;const Separator:string;const Index:Integer;const Value:string)
13423:  Function ListItemIndex( const List, Separator, Item : string) : Integer
13424:  Function SystemTObjectInstance : LongWord
13425:  Function IsCompiledWithPackages : Boolean
13426:  Function JJclGUIDToString( const GUID : TGUID) : string
13427:  Function JJclStringToGUID( const S : string) : TGUID
13428:   SIRegister_TJclIntfCriticalSection(CL);
13429:   SIRegister_TJclSimpleLog(CL);
13430:  Procedure InitSimpleLog( const ALogFileName : string)
13431: end;
13432:
13433: procedure SIRegister_JclBorlandTools(CL: TPSPascalCompiler);
13434: begin
13435:   FindClass('TOBJECT'),'EJclBorRADException
```

```
13436:   TJclBorRADToolKind', '( brDelphi, brCppBuilder, brBorlandDevStudio )
13437:   TJclBorRADToolEdition', '( deOPEN, dePRO, deSVR )
13438:   TJclBorRADToolEdition', '( deSTD, dePRO, deCSS, deARC )
13439:   TJclBorRADToolPath', 'string
13440:  'SupportedDelphiVersions','LongInt').SetInt( 5 or  6 or  7 or  8 or  9 or  10 or  11);
13441:  'SupportedBCBVersions','LongInt').SetInt( 5 or  6 or  10 or  11);
13442:  'SupportedBDSVersions','LongInt').SetInt( 1 or  2 or  3 or  4 or  5);
13443:  BorRADToolRepositoryPagesSection','String').SetString( 'Repository Pages
13444:  BorRADToolRepositoryDialogsPage','String').SetString( 'Dialogs
13445:  BorRADToolRepositoryFormsPage','String').SetString( 'Forms
13446:  BorRADToolRepositoryProjectsPage','String').SetString( 'Projects
13447:  BorRADToolRepositoryDataModulesPage','String').SetString( 'Data Modules
13448:  BorRADToolRepositoryObjectType','String').SetString( 'Type
13449:  BorRADToolRepositoryFormTemplate','String').SetString( 'FormTemplate
13450:  BorRADToolRepositoryProjectTemplate','String').SetString( 'ProjectTemplate
13451:  BorRADToolRepositoryObjectName','String').SetString( 'Name
13452:  BorRADToolRepositoryObjectPage','String').SetString( 'Page
13453:  BorRADToolRepositoryObjectIcon','String').SetString( 'Icon
13454:  BorRADToolRepositoryObjectDescr','String').SetString( 'Description
13455:  BorRADToolRepositoryObjectAuthor','String').SetString( 'Author
13456:  BorRADToolRepositoryObjectAncestor','String').SetString( 'Ancestor
13457:  BorRADToolRepositoryObjectDesigner','String').SetString( 'Designer
13458:  BorRADToolRepositoryDesignerDfm','String').SetString( 'dfm
13459:  BorRADToolRepositoryDesignerXfm','String').SetString( 'xfm
13460:  BorRADToolRepositoryObjectNewForm','String').SetString( 'DefaultNewForm
13461:  BorRADToolRepositoryObjectMainForm','String').SetString( 'DefaultMainForm
13462:  SourceExtensionDelphiPackage','String').SetString( '.dpk
13463:  SourceExtensionBCBPackage','String').SetString( '.bpk
13464:  SourceExtensionDelphiProject','String').SetString( '.dpr
13465:  SourceExtensionBCBProject','String').SetString( '.bpr
13466:  SourceExtensionBDSProject','String').SetString( '.bdsproj
13467:  SourceExtensionDProject','String').SetString( '.dproj
13468:  BinaryExtensionPackage','String').SetString( '.bpl
13469:  BinaryExtensionLibrary','String').SetString( '.dll
13470:  BinaryExtensionExecutable','String').SetString( '.exe
13471:  CompilerExtensionDCP','String').SetString( '.dcp
13472:  CompilerExtensionBPI','String').SetString( '.bpi
13473:  CompilerExtensionLIB','String').SetString( '.lib
13474:  CompilerExtensionTDS','String').SetString( '.tds
13475:  CompilerExtensionMAP','String').SetString( '.map
13476:  CompilerExtensionDRC','String').SetString( '.drc
13477:  CompilerExtensionDEF','String').SetString( '.def
13478:  SourceExtensionCPP','String').SetString( '.cpp
13479:  SourceExtensionH','String').SetString( '.h
13480:  SourceExtensionPAS','String').SetString( '.pas
13481:  SourceExtensionDFM','String').SetString( '.dfm
13482:  SourceExtensionXFM','String').SetString( '.xfm
13483:  SourceDescriptionPAS','String').SetString( 'Pascal source file
13484:  SourceDescriptionCPP','String').SetString( 'C++ source file
13485:  DesignerVCL','String').SetString( 'VCL
13486:  DesignerCLX','String').SetString( 'CLX
13487:  ProjectTypePackage','String').SetString( 'package
13488:  ProjectTypeLibrary','String').SetString( 'library
13489:  ProjectTypeProgram','String').SetString( 'program
13490:  Personality32Bit','String').SetString( '32 bit
13491:  Personality64Bit','String').SetString( '64 bit
13492:  PersonalityDelphi','String').SetString( 'Delphi
13493:  PersonalityDelphiDotNet','String').SetString( 'Delphi.net
13494:  PersonalityBCB','String').SetString( 'C++Builder
13495:  PersonalityCSB','String').SetString( 'C#Builder
13496:  PersonalityVB','String').SetString( 'Visual Basic
13497:  PersonalityDesign','String').SetString( 'Design
13498:  PersonalityUnknown','String').SetString( 'Unknown personality
13499:  PersonalityBDS','String').SetString( 'Borland Developer Studio
13500:  DOFDirectoriesSection','String').SetString( 'Directories
13501:  DOFUnitOutputDirKey','String').SetString( 'UnitOutputDir
13502:  DOFSearchPathName','String').SetString( 'SearchPath
13503:  DOFConditionals','String').SetString( 'Conditionals
13504:  DOFLinkerSection','String').SetString( 'Linker
13505:  DOFPackagesKey','String').SetString( 'Packages
13506:  DOFCompilerSection','String').SetString( 'Compiler
13507:  DOFPackageNoLinkKey','String').SetString( 'PackageNoLink
13508:  DOFAdditionalSection','String').SetString( 'Additional
13509:  DOFOptionsKey','String').SetString( 'Options
13510:   TJclBorPersonality', '( bpDelphi32, bpDelphi64, bpBCBuilder32, b'
13511:    +'pBCBuilder64, bpDelphiNet32, bpDelphiNet64, bpCSBuilder32, bpCSBuilder64, '
13512:    +'bpVisualBasic32, bpVisualBasic64, bpDesign, bpUnknown )
13513:   TJclBorPersonalities', 'set of TJclBorPersonality
13514:   TJclBorDesigner', '( bdVCL, bdCLX )
13515:   TJclBorDesigners', 'set of TJclBorDesigner
13516:   TJclBorPlatform', '( bp32bit, bp64bit )
13517:   FindClass('TOBJECT'),'TJclBorRADToolInstallation
13518:   SIRegister_TJclBorRADToolInstallationObject(CL);
13519:   SIRegister_TJclBorlandOpenHelp(CL);
13520:   TJclHelp2Object', '( hoRegisterSession, hoRegister, hoPlugin )
13521:   TJclHelp2Objects', 'set of TJclHelp2Object
13522:   SIRegister_TJclHelp2Manager(CL);
13523:   SIRegister_TJclBorRADToolIdeTool(CL);
13524:   SIRegister_TJclBorRADToolIdePackages(CL);
```

```
13525:   SIRegister_IJclCommandLineTool(CL);
13526:   FindClass('TOBJECT'),'EJclCommandLineToolError
13527:   SIRegister_TJclCommandLineTool(CL);
13528:   SIRegister_TJclBorlandCommandLineTool(CL);
13529:   SIRegister_TJclBCC32(CL);
13530:   SIRegister_TJclDCC32(CL);
13531:   TJclDCC', 'TJclDCC32
13532:   SIRegister_TJclBpr2Mak(CL);
13533:   SIRegister_TJclBorlandMake(CL);
13534:   SIRegister_TJclBorRADToolPalette(CL);
13535:   SIRegister_TJclBorRADToolRepository(CL);
13536:   TCommandLineTool', '( clAsm, clBcc32, clDcc32, clDccIL, clMake,clProj2Mak )
13537:   TCommandLineTools', 'set of TCommandLineTool
13538:   //TJclBorRADToolInstallationClass', 'class of TJclBorRADToolInstallation
13539:   SIRegister_TJclBorRADToolInstallation(CL);
13540:   SIRegister_TJclBCBInstallation(CL);
13541:   SIRegister_TJclDelphiInstallation(CL);
13542:   SIRegister_TJclDCCIL(CL);
13543:   SIRegister_TJclBDSInstallation(CL);
13544:   TTraverseMethod', 'Function ( Installation : TJclBorRADToolInstallation) : Boolean
13545:   SIRegister_TJclBorRADToolInstallations(CL);
13546:  Function BPLFileName( const BPLPath, PackageFileName : string) : string
13547:  Function BinaryFileName( const OutputPath, ProjectFileName : string) : string
13548:  Function IsDelphiPackage( const FileName : string) : Boolean
13549:  Function IsDelphiProject( const FileName : string) : Boolean
13550:  Function IsBCBPackage( const FileName : string) : Boolean
13551:  Function IsBCBProject( const FileName : string) : Boolean
13552:  Procedure GetDPRFileInfo(const DPRFileName:string;out BinaryExtensio:string;const LibSuffx:PString);
13553:  Procedure GetBPRFileInfo(const BPRFileName:string;out BinaryFileName:string;const Descript:PString);
13554:  Procedure GetDPKFileInfo(const DPKFileName:string;out RunOnly:Bool;const LibSuffix:PString;const
       Descript:PString;
13555:  Procedure GetBPKFileInfo(const BPKFileName:string;out RunOnly:Bool;const BinaryFName:PString;const
       Descript:PString);
13556:  function SamePath(const Path1, Path2: string): Boolean;
13557: end;
13558:
13559: procedure SIRegister_JclFileUtils_max(CL: TPSPascalCompiler);
13560: begin
13561:  'ERROR_NO_MORE_FILES','LongInt').SetInt( 18);
13562:  //Function stat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13563:  //Function fstat64( FileDes: Integer;var StatBuffer : TStatBuf64) : Integer
13564:  //Function lstat64( FileName: PChar;var StatBuffer : TStatBuf64) : Integer
13565:  'LPathSeparator','String').SetString( '/
13566:  'LDirDelimiter','String').SetString( '/
13567:  'LDirSeparator','String').SetString( ':
13568:  'JXPathDevicePrefix','String').SetString( '\\.\
13569:  'JXPathSeparator','String').SetString( '\
13570:  'JXDirDelimiter','String').SetString( '\
13571:  'JXDirSeparator','String').SetString( ';
13572:  'JXPathUncPrefix','String').SetString( '\\
13573:  'faNormalFile','LongWord').SetUInt( $00000080);
13574:  //'faUnixSpecific','').SetString( faSymLink);
13575:   JXTCompactPath', '( cpCenter, cpEnd )
13576:     _WIN32_FILE_ATTRIBUTE_DATA', 'record dwFileAttributes : DWORD; f'
13577:    +'tCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime :'
13578:    +' TFileTime; nFileSizeHigh : DWORD; nFileSizeLow : DWORD; end
13579:   TWin32FileAttributeData', '_WIN32_FILE_ATTRIBUTE_DATA
13580:   WIN32_FILE_ATTRIBUTE_DATA', '_WIN32_FILE_ATTRIBUTE_DATA
13581:
13582:  Function jxPathAddSeparator( const Path : string) : string
13583:  Function jxPathAddExtension( const Path, Extension : string) : string
13584:  Function jxPathAppend( const Path, Append : string) : string
13585:  Function jxPathBuildRoot( const Drive : Byte) : string
13586:  Function jxPathCanonicalize( const Path : string) : string
13587:  Function jxPathCommonPrefix( const Path1, Path2 : string) : Integer
13588:  Function jxPathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string
13589:  Procedure jxPathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
13590:  Function jxPathExtractFileDirFixed( const S : string) : string
13591:  Function jxPathExtractFileNameNoExt( const Path : string) : string
13592:  Function jxPathExtractPathDepth( const Path : string; Depth : Integer) : string
13593:  Function jxPathGetDepth( const Path : string) : Integer
13594:  Function jxPathGetLongName( const Path : string) : string
13595:  Function jxPathGetShortName( const Path : string) : string
13596:  Function jxPathGetLongName( const Path : string) : string
13597:  Function jxPathGetShortName( const Path : string) : string
13598:  Function jxPathGetRelativePath( Origin, Destination : string) : string
13599:  Function jxPathGetTempPath : string
13600:  Function jxPathIsAbsolute( const Path : string) : Boolean
13601:  Function jxPathIsChild( const Path, Base : string) : Boolean
13602:  Function jxPathIsDiskDevice( const Path : string) : Boolean
13603:  Function jxPathIsUNC( const Path : string) : Boolean
13604:  Function jxPathRemoveSeparator( const Path : string) : string
13605:  Function jxPathRemoveExtension( const Path : string) : string
13606:  Function jxPathGetPhysicalPath( const LocalizedPath : string) : string
13607:  Function jxPathGetLocalizedPath( const PhysicalPath : string) : string
13608:   JxTFileListOption', '( flFullNames, flRecursive, flMaskedSubfolders)
13609:   JxTFileListOptions', 'set of TFileListOption
13610:   JxTJclAttributeMatch', '( amAny, amExact, amSubSetOf, amSuperSetOf, amCustom )
13611:   TFileHandler', 'Procedure ( const FileName : string)
```

```
13612:   TFileHandlerEx', 'Procedure ( const Directory : string; const FileInfo : TSearchRec)
13613:  Function BuildFileList( const Path : string; const Attr : Integer; const List : TStrings) : Boolean
13614:  //Function AdvBuildFileList( const Path : string; const Attr : Integer; const Files : TStrings; const
       AttributeMatch : TJclAttributeMatch;const Optis:TFileListOptions;const SubfoldersMask:string;const
       FileMatchFunc:TFileMatchFunc):Bool;
13615:  Function jxVerifyFileAttributeMask( var RejectedAttributes, RequiredAttributes : Int) : Boolean
13616:  Function jxIsFileAttributeMatch(FileAttributes,RejectedAttributes,RequiredAttributes:Int):Boolean;
13617:  Function jxFileAttributesStr( const FileInfo : TSearchRec) : string
13618:  Function jxIsFileNameMatch(FileName:string;const Mask:string;const CaseSensitive:Boolean):Boolean;
13619:  Procedure jxEnumFiles(const Path:string; HandleFile:TFileHandlerEx;
       RejectedAttributes:Integer;RequiredAttributes : Integer; Abort : TObject)
13620:  Procedure jxEnumDirectories(const Root:string;const HandleDirectory:TFileHandler;const
       IncludeHiddenDirects:Boolean; const SubDirectoriesMask : string; Abort : TObject; ResolveSymLinks :
       Boolean)
13621:  Procedure jxCreateEmptyFile( const FileName : string)
13622:  Function jxCloseVolume( var Volume : THandle) : Boolean
13623:  Function jxDeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean) : Boolean
13624:  Function jxCopyDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13625:  Function jxMoveDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13626:  Function jxDelTree( const Path : string) : Boolean
13627:  //Function DelTreeEx(const Path:string;AbortOnFailure:Boolean; Progress:TDelTreeProgress):Boolean
13628:  Function jxDiskInDrive( Drive : Char) : Boolean
13629:  Function jxDirectoryExists( const Name : string; ResolveSymLinks : Boolean) : Boolean
13630:  Function jxFileCreateTemp( var Prefix : string) : THandle
13631:  Function jxFileBackup( const FileName : string; Move : Boolean) : Boolean
13632:  Function jxFileCopy( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13633:  Function jxFileDelete( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13634:  Function jxFileExists( const FileName : string) : Boolean
13635:  Function jxFileMove( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13636:  Function jxFileRestore( const FileName : string) : Boolean
13637:  Function jxGetBackupFileName( const FileName : string) : string
13638:  Function jxIsBackupFileName( const FileName : string) : Boolean
13639:  Function jxFileGetDisplayName( const FileName : string) : string
13640:  Function jxFileGetGroupName( const FileName : string; ResolveSymLinks : Boolean) : string
13641:  Function jxFileGetOwnerName( const FileName : string; ResolveSymLinks : Boolean) : string
13642:  Function jxFileGetSize( const FileName : string) : Int64
13643:  Function jxFileGetTempName( const Prefix : string) : string
13644:  Function jxFileGetTypeName( const FileName : string) : string
13645:  Function jxFindUnusedFileName(FileName:string; const FileExt : string; NumberPrefix : string) : string
13646:  Function jxForceDirectories( Name : string) : Boolean
13647:  Function jxGetDirectorySize( const Path : string) : Int64
13648:  Function jxGetDriveTypeStr( const Drive : Char) : string
13649:  Function jxGetFileAgeCoherence( const FileName : string) : Boolean
13650:  Procedure jxGetFileAttributeList( const Items : TStrings; const Attr : Integer)
13651:  Procedure jxGetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
13652:  Function jxGetFileInformation( const FileName : string; out FileInfo : TSearchRec) : Boolean;
13653:  Function jxGetFileInformation1( const FileName : string) : TSearchRec;
13654:  //Function GetFileStatus(const FileName:string;out StatBuf:TStatBuf64;const
       ResolveSymLinks:Boolean):Integer
13655:  Function jxGetFileLastWrite( const FName : string) : TFileTime;
13656:  Function jxGetFileLastWrite1( const FName : string; out LocalTime : TDateTime) : Boolean;
13657:  Function jxGetFileLastAccess( const FName : string) : TFileTime;
13658:  Function jxGetFileLastAccess1( const FName : string; out LocalTime : TDateTime) : Boolean;
13659:  Function jxGetFileCreation( const FName : string) : TFileTime;
13660:  Function jxGetFileCreation1( const FName : string; out LocalTime : TDateTime) : Boolean;
13661:  Function jxGetFileLastWrite( const FName : string;out TimeStamp:Integer;ResolveSymLinks : Bool):Bool;
13662:  Function jxGetFileLastWrite1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool): Bool;
13663:  Function jxGetFileLastWrite2( const FName : string; ResolveSymLinks : Boolean) : Integer;
13664:  Function jxGetFileLastAccess(const FName:string; out TimeStamp:Integer;ResolveSymLinks: Bool): Bool;
13665:  Function jxGetFileLastAccess1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13666:  Function jxGetFileLastAccess2(const FName:string; ResolveSymLinks:Boolean): Integer;
13667:  Function jxGetFileLastAttrChange(const FName:string;out TimeStamp:Integer;ResolveSymLinks:Bool): Bool;
13668:  Function jxGetFileLastAttrChange1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13669:  Function jxGetFileLastAttrChange2( const FName : string; ResolveSymLinks:Boolean): Integer;
13670:  Function jxGetModulePath( const Module : HMODULE) : string
13671:  Function jxGetSizeOfFile( const FileName : string) : Int64;
13672:  Function jxGetSizeOfFile1( const FileInfo : TSearchRec) : Int64;
13673:  Function jxGetSizeOfFile2( Handle : THandle) : Int64;
13674:  Function jxGetStandardFileInfo( const FileName : string) : TWin32FileAttributeData
13675:  Function jxIsDirectory( const FileName : string; ResolveSymLinks : Boolean) : Boolean
13676:  Function jxIsRootDirectory( const CanonicFileName : string) : Boolean
13677:  Function jxLockVolume( const Volume : string; var Handle : THandle) : Boolean
13678:  Function jxOpenVolume( const Drive : Char) : THandle
13679:  Function jxSetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
13680:  Function jxSetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
13681:  Function jxSetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
13682:  Function jxSetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
13683:  Function jxSetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
13684:  Function jxSetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
13685:  Procedure jxShredFile( const FileName : string; Times : Integer)
13686:  Function jxUnlockVolume( var Handle : THandle) : Boolean
13687:  Function jxCreateSymbolicLink( const Name, Target : string) : Boolean
13688:  Function jxSymbolicLinkTarget( const Name : string) : string
13689:   TAttributeInterest', '( aiIgnored, aiRejected, aiRequired )
13690:   SIRegister_TJclCustomFileAttrMask(CL);
13691:   SIRegister_TJclFileAttributeMask(CL);
13692:   TFileSearchOption', '( fsIncludeSubDirectories, fsIncludeHiddenS'
13693:    +'ubDirectories, fsLastChangeAfter, fsLastChangeBefore, fsMaxSize, fsMinSize)
13694:   TFileSearchOptions', 'set of TFileSearchOption
```

```
13695:   TFileSearchTaskID', 'Integer
13696:   TFileSearchTerminationEvent', 'Procedure ( const ID : TFileSearc'
13697:    +'hTaskID; const Aborted : Boolean)
13698:   TFileEnumeratorSyncMode', '( smPerFile, smPerDirectory )
13699:   SIRegister_IJclFileEnumerator(CL);
13700:   SIRegister_TJclFileEnumerator(CL);
13701:  Function JxFileSearch : IJclFileEnumerator
13702:   JxTFileFlag', '( ffDebug, ffInfoInferred, ffPatched,ffPreRelease,ffPrivateBuild, ffSpecialBuild )
13703:   JxTFileFlags', 'set of TFileFlag
13704:   FindClass('TOBJECT'),'EJclFileVersionInfoError
13705:   SIRegister_TJclFileVersionInfo(CL);
13706:  Function jxOSIdentToString( const OSIdent : DWORD) : string
13707:  Function jxOSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
13708:  Function jxVersionResourceAvailable( const FileName : string) : Boolean
13709:   TFileVersionFormat', '( vfMajorMinor, vfFull )
13710:  Function jxFormatVersionString( const HiV, LoV : Word) : string;
13711:  Function jxFormatVersionString1( const Major, Minor, Build, Revision : Word) : string;
13712:  //Function FormatVersionString2( const FixedInfo : TVSFixedFileInfo;VersionFormat:TFileVersionFormat):str;
13713:  //Procedure VersionExtractFileInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,Revision:Word);
13714:  //Procedure VersionExtractProductInfo(const FixedInfo:TVSFixedFileInfo;var Major,Minor,Build,
      Revision:Word);
13715:  //Function VersionFixedFileInfo( const FileName : string; var FixedInfo : TVSFixedFileInfo) : Boolean
13716:  Function jxVersionFixedFileInfoString( const FileName : string; VersionFormat : TFileVersionFormat; const
      NotAvailableText : string) : string
13717:   SIRegister_TJclTempFileStream(CL);
13718:   FindClass('TOBJECT'),'TJclCustomFileMapping
13719:   SIRegister_TJclFileMappingView(CL);
13720:   TJclFileMappingRoundOffset', '( rvDown, rvUp )
13721:   SIRegister_TJclCustomFileMapping(CL);
13722:   SIRegister_TJclFileMapping(CL);
13723:   SIRegister_TJclSwapFileMapping(CL);
13724:   SIRegister_TJclFileMappingStream(CL);
13725:   TJclMappedTextReaderIndex', '( tiNoIndex, tiFull )
13726:   //PPCharArray', '^TPCharArray // will not work
13727:   SIRegister_TJclMappedTextReader(CL);
13728:   SIRegister_TJclFileMaskComparator(CL);
13729:   FindClass('TOBJECT'),'EJclPathError
13730:   FindClass('TOBJECT'),'EJclFileUtilsError
13731:   FindClass('TOBJECT'),'EJclTempFileStreamError
13732:   FindClass('TOBJECT'),'EJclTempFileStreamError
13733:   FindClass('TOBJECT'),'EJclFileMappingError
13734:   FindClass('TOBJECT'),'EJclFileMappingViewError
13735:  Function jxPathGetLongName2( const Path : string) : string
13736:  Function jxWin32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13737:  Function jxWin32MoveFileReplaceExisting( const SrcFileName, DstFileName : string) : Boolean
13738:  Function jxWin32BackupFile( const FileName : string; Move : Boolean) : Boolean
13739:  Function jxWin32RestoreFile( const FileName : string) : Boolean
13740:  Function jxSamePath( const Path1, Path2 : string) : Boolean
13741:  Procedure jxPathListAddItems( var List : string; const Items : string)
13742:  Procedure jxPathListIncludeItems( var List : string; const Items : string)
13743:  Procedure jxPathListDelItems( var List : string; const Items : string)
13744:  Procedure jxPathListDelItem( var List : string; const Index : Integer)
13745:  Function jxPathListItemCount( const List : string) : Integer
13746:  Function jxPathListGetItem( const List : string; const Index : Integer) : string
13747:  Procedure jxPathListSetItem( var List : string; const Index : Integer; const Value : string)
13748:  Function jxPathListItemIndex( const List, Item : string) : Integer
13749:  Function jxParamName(Idx:Int;const Separator:string;const AllowedPrefixChars:string;TrimName:Bool):string;
13750:  Function jxParamValue(Index : Integer; const Separator : string; TrimValue : Boolean) : string;
13751:  Function jxParamValue1(const SearchName:string; const Separator : string; CaseSensitive : Boolean; const
      AllowedPrefixCharacters : string; TrimValue : Boolean) : string;
13752:  Function jxParamPos( const SearchName : string; const Separator : string; CaseSensitive : Boolean; const
      AllowedPrefixCharacters : string) : Integer
13753: end;
13754:
13755: procedure SIRegister_FileUtil(CL: TPSPascalCompiler);
13756: begin
13757:  'UTF8FileHeader','String').SetString( #$ef#$bb#$bf);
13758:  Function lCompareFilenames( const Filename1, Filename2 : string) : integer
13759:  Function lCompareFilenamesIgnoreCase( const Filename1, Filename2 : string) : integer
13760:  Function lCompareFilenames( const Filename1, Filename2 : string; ResolveLinks : boolean) : integer
13761:  Function lCompareFilenames(Filename1:PChar;Len1:int;Filename2:PChar;Len2:int;ResolveLinks:boolean):int;
13762:  Function lFilenameIsAbsolute( const TheFilename : string) : boolean
13763:  Function lFilenameIsWinAbsolute( const TheFilename : string) : boolean
13764:  Function lFilenameIsUnixAbsolute( const TheFilename : string) : boolean
13765:  Procedure lCheckIfFileIsExecutable( const AFilename : string)
13766:  Procedure lCheckIfFileIsSymlink( const AFilename : string)
13767:  Function lFileIsReadable( const AFilename : string) : boolean
13768:  Function lFileIsWritable( const AFilename : string) : boolean
13769:  Function lFileIsText( const AFilename : string) : boolean
13770:  Function lFileIsText( const AFilename : string; out FileReadable : boolean) : boolean
13771:  Function lFileIsExecutable( const AFilename : string) : boolean
13772:  Function lFileIsSymlink( const AFilename : string) : boolean
13773:  Function lFileIsHardLink( const AFilename : string) : boolean
13774:  Function lFileSize( const Filename : string) : int64;
13775:  Function lGetFileDescription( const AFilename : string) : string
13776:  Function lReadAllLinks( const Filename : string; ExceptionOnError : boolean) : string
13777:  Function lTryReadAllLinks( const Filename : string) : string
13778:  Function lDirPathExists( const FileName : String) : Boolean
13779:  Function lForceDirectory( DirectoryName : string) : boolean
```

```
13780:  Function lDeleteDirectory( const DirectoryName : string; OnlyChildren : boolean) : boolean
13781:  Function lProgramDirectory : string
13782:  Function lDirectoryIsWritable( const DirectoryName : string) : boolean
13783:  Function lExtractFileNameOnly( const AFilename : string) : string
13784:  Function lExtractFileNameWithoutExt( const AFilename : string) : string
13785:  Function lCompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
13786:  Function lCompareFileExt( const Filename, Ext : string) : integer;
13787:  Function lFilenameIsPascalUnit( const Filename : string) : boolean
13788:  Function lAppendPathDelim( const Path : string) : string
13789:  Function lChompPathDelim( const Path : string) : string
13790:  Function lTrimFilename( const AFilename : string) : string
13791:  Function lCleanAndExpandFilename( const Filename : string) : string
13792:  Function lCleanAndExpandDirectory( const Filename : string) : string
13793:  Function lCreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
13794:  Function lCreateRelativePath( const Filename, BaseDirectory : string; UsePointDirectory : boolean;
      AlwaysRequireSharedBaseFolder : Boolean) : string
13795:  Function lCreateAbsolutePath( const Filename, BaseDirectory : string) : string
13796:  Function lFileIsInPath( const Filename, Path : string) : boolean
13797:  Function lFileIsInDirectory( const Filename, Directory : string) : boolean
13798:   TSearchFileInPathFlag', '( sffDontSearchInBasePath, sffSearchLoUpCase )
13799:   TSearchFileInPathFlags', 'set of TSearchFileInPathFlag
13800:  'AllDirectoryEntriesMask','String').SetString( '*
13801:  Function lGetAllFilesMask : string
13802:  Function lGetExeExt : string
13803:  Function lSearchFileInPath( const Filename, BasePath, SearchPath, Delimiter : string; Flags :
      TSearchFileInPathFlags) : string
13804:  Function lSearchAllFilesInPath( const Filename, BasePath, SearchPath, Delimiter:string;Flags :
      TSearchFileInPathFlags) : TStrings
13805:  Function lFindDiskFilename( const Filename : string) : string
13806:  Function lFindDiskFileCaseInsensitive( const Filename : string) : string
13807:  Function lFindDefaultExecutablePath( const Executable : string; const BaseDir: string):string
13808:  Function lGetDarwinSystemFilename( Filename : string) : string
13809:   SIRegister_TFileIterator(CL);
13810:   TFileFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13811:   TDirectoryFoundEvent', 'Procedure ( FileIterator : TFileIterator)
13812:   TDirectoryEnterEvent', 'Procedure ( FileIterator : TFileIterator)
13813:   SIRegister_TFileSearcher(CL);
13814:  Function lFindAllFiles(const SearchPath:String;SearchMsk:String;SearchSubDirs:Bool):TStringList
13815:  Function lFindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
13816:  // TCopyFileFlag', '( cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime )
13817:  // TCopyFileFlags', 'set of TCopyFileFlag
13818:  Function lCopyFile( const SrcFilename, DestFilename : string; Flags : TCopyFileFlags) : boolean
13819:  Function lCopyFile( const SrcFilename, DestFilename : string; PreserveTime : boolean) : boolean
13820:  Function lCopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
13821:  Function lReadFileToString( const Filename : string) : string
13822:  Function lGetTempFilename( const Directory, Prefix : string) : string
13823:  {Function NeedRTLAnsi : boolean
13824:  Procedure SetNeedRTLAnsi( NewValue : boolean)
13825:  Function UTF8ToSys( const s : string) : string
13826:  Function SysToUTF8( const s : string) : string
13827:  Function ConsoleToUTF8( const s : string) : string
13828:  Function UTF8ToConsole( const s : string) : string}
13829:  Function FileExistsUTF8( const Filename : string) : boolean
13830:  Function FileAgeUTF8( const FileName : string) : Longint
13831:  Function DirectoryExistsUTF8( const Directory : string) : Boolean
13832:  Function ExpandFileNameUTF8( const FileName : string) : string
13833:  Function ExpandUNCFileNameUTF8( const FileName : string) : string
13834:  Function ExtractShortPathNameUTF8( const FileName : String) : String
13835:  Function FindFirstUTF8( const Path : string; Attr : Longint; out Rslt : TSearchRec) : Longint
13836:  Function FindNextUTF8( var Rslt : TSearchRec) : Longint
13837:  Procedure FindCloseUTF8( var F : TSearchrec)
13838:  Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
13839:  Function FileGetAttrUTF8( const FileName : String) : Longint
13840:  Function FileSetAttrUTF8( const Filename : String; Attr : longint) : Longint
13841:  Function DeleteFileUTF8( const FileName : String) : Boolean
13842:  Function RenameFileUTF8( const OldName, NewName : String) : Boolean
13843:  Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
13844:  Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
13845:  Function GetCurrentDirUTF8 : String
13846:  Function SetCurrentDirUTF8( const NewDir : String) : Boolean
13847:  Function CreateDirUTF8( const NewDir : String) : Boolean
13848:  Function RemoveDirUTF8( const Dir : String) : Boolean
13849:  Function ForceDirectoriesUTF8( const Dir : string) : Boolean
13850:  Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
13851:  Function FileCreateUTF8( const FileName : string) : THandle;
13852:  Function FileCreateUTF81( const FileName : string; Rights : Cardinal) : THandle;
13853:  Function ParamStrUTF8( Param : Integer) : string
13854:  Function GetEnvironmentStringUTF8( Index : Integer) : string
13855:  Function GetEnvironmentVariableUTF8( const EnvVar : string) : String
13856:  Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
13857:  Function GetAppConfigFileUTF8(Global:Boolean; SubDir:boolean; CreateDir : boolean) : string
13858:  Function SysErrorMessageUTF8( ErrorCode : Integer) : String
13859: end;
13860:
13861: procedure SIRegister_Keyboard(CL: TPSPascalCompiler);
13862: begin
13863:   //VK_F23 = 134;
13864:   //{$EXTERNALSYM VK_F24}
13865:   //VK_F24 = 135;
```

```
13866:  TVirtualKeyCode', 'Integer
13867:  'VK_MOUSEWHEELUP','integer').SetInt(134);
13868:  'VK_MOUSEWHEELDOWN','integer').SetInt(135);
13869:  Function glIsKeyDown( c : Char) : Boolean;
13870:  Function glIsKeyDown1( vk : TVirtualKeyCode) : Boolean;
13871:  Function glKeyPressed( minVkCode : TVirtualKeyCode) : TVirtualKeyCode
13872:  Function glVirtualKeyCodeToKeyName( vk : TVirtualKeyCode) : String
13873:  Function glKeyNameToVirtualKeyCode( const keyName : String) : TVirtualKeyCode
13874:  Function glCharToVirtualKeyCode( c : Char) : TVirtualKeyCode
13875:  Procedure glKeyboardNotifyWheelMoved( wheelDelta : Integer)
13876: end;
13877:
13878: procedure SIRegister_GLCrossPlatform(CL: TPSPascalCompiler);
13879: begin
13880:   TGLPoint', 'TPoint
13881:   //PGLPoint', '^TGLPoint // will not work
13882:   TGLRect', 'TRect
13883:   //PGLRect', '^TGLRect // will not work
13884:   TDelphiColor', 'TColor
13885:   TGLPicture', 'TPicture
13886:   TGLGraphic', 'TGraphic
13887:   TGLBitmap', 'TBitmap
13888:   //TGraphicClass', 'class of TGraphic
13889:   TGLTextLayout', '( tlTop, tlCenter, tlBottom )
13890:   TGLMouseButton', '( mbLeft, mbRight, mbMiddle )
13891:   TGLMouseEvent', 'Procedure ( Sender : TObject; Button : TGLMouse'
13892:    +'Button; Shift : TShiftState; X, Y : Integer)
13893:   TGLMouseMoveEvent', 'TMouseMoveEvent
13894:   TGLKeyEvent', 'TKeyEvent
13895:   TGLKeyPressEvent', 'TKeyPressEvent
13896:   EGLOSError', 'EWin32Error
13897:   EGLOSError', 'EWin32Error
13898:   EGLOSError', 'EOSError
13899:  'glsAllFilter','string').SetString('All // sAllFilter
13900:  Function GLPoint( const x, y : Integer) : TGLPoint
13901:  Function GLRGB( const r, g, b : Byte) : TColor
13902:  Function GLColorToRGB( color : TColor) : TColor
13903:  Function GLGetRValue( rgb : DWORD) : Byte
13904:  Function GLGetGValue( rgb : DWORD) : Byte
13905:  Function GLGetBValue( rgb : DWORD) : Byte
13906:  Procedure GLInitWinColors
13907:  Function GLRect( const aLeft, aTop, aRight, aBottom : Integer) : TGLRect
13908:  Procedure GLInflateGLRect( var aRect : TGLRect; dx, dy : Integer)
13909:  Procedure GLIntersectGLRect( var aRect : TGLRect; const rect2 : TGLRect)
13910:  Procedure GLInformationDlg( const msg : String)
13911:  Function GLQuestionDlg( const msg : String) : Boolean
13912:  Function GLInputDlg( const aCaption, aPrompt, aDefault : String) : String
13913:  Function GLSavePictureDialog( var aFileName : String; const aTitle : String) : Boolean
13914:  Function GLOpenPictureDialog( var aFileName : String; const aTitle : String) : Boolean
13915:  Function GLApplicationTerminated : Boolean
13916:  Procedure GLRaiseLastOSError
13917:  Procedure GLFreeAndNil( var anObject: TObject)
13918:  Function GLGetDeviceLogicalPixelsX( device : Cardinal) : Integer
13919:  Function GLGetCurrentColorDepth : Integer
13920:  Function GLPixelFormatToColorBits( aPixelFormat : TPixelFormat) : Integer
13921:  Function GLBitmapScanLine( aBitmap : TGLBitmap; aRow : Integer) : Pointer
13922:  Procedure GLSleep( length : Cardinal)
13923:  Procedure GLQueryPerformanceCounter( var val : Int64)
13924:  Function GLQueryPerformanceFrequency( var val : Int64) : Boolean
13925:  Function GLStartPrecisionTimer : Int64
13926:  Function GLPrecisionTimerLap( const precisionTimer : Int64) : Double
13927:  Function GLStopPrecisionTimer( const precisionTimer : Int64) : Double
13928:  Function GLRDTSC : Int64
13929:  Procedure GLLoadBitmapFromInstance( ABitmap : TBitmap; AName : string)
13930:  Function GLOKMessageBox( const Text, Caption : string) : Integer
13931:  Procedure GLShowHTMLUrl( Url : String)
13932:  Procedure GLShowCursor( AShow : boolean)
13933:  Procedure GLSetCursorPos( AScreenX, AScreenY : integer)
13934:  Procedure GLGetCursorPos( var point : TGLPoint)
13935:  Function GLGetScreenWidth : integer
13936:  Function GLGetScreenHeight : integer
13937:  Function GLGetTickCount : int64
13938:  function RemoveSpaces(const str : String) : String;
13939:    TNormalMapSpace', '( nmsObject, nmsTangent )
13940:  Procedure CalcObjectSpaceLightVectors(Light:TAffineVector;Vertices TAffineVectorList;Colors:TVectorList)
13941:  Procedure SetupTangentSpace( Vertices, Normals, TexCoords, Tangents, BiNormals : TAffineVectorList)
13942:  Procedure CalcTangentSpaceLightVectors( Light : TAffineVector; Vertices, Normals, Tangents, BiNormals :
       TAffineVectorList; Colors : TVectorList)
13943:  Function CreateObjectSpaceNormalMap( Width,Height:Integer;HiNormals,
       HiTexCoords:TAffineVectorList):TGLBitmap
13944:  Function CreateTangentSpaceNormalMap( Width, Height : Integer; HiNormals, HiTexCoords, LoNormals,
       LoTexCoords, Tangents, BiNormals : TAffineVectorList) : TGLBitmap
13945: end;
13946:
13947: procedure SIRegister_GLStarRecord(CL: TPSPascalCompiler);
13948: begin
13949:   TGLStarRecord', 'record RA: Word; DEC: SmallInt; BVColorIndex: Byte; VMagnitude: Byte; end
13950:  // PGLStarRecord', '^TGLStarRecord // will not work
13951:  Function StarRecordPositionZUp( const starRecord : TGLStarRecord) : TAffineVector
```

```
13952:  Function StarRecordPositionYUp( const starRecord : TGLStarRecord) : TAffineVector
13953:  Function StarRecordColor( const starRecord : TGLStarRecord; bias : Single) : TVector
13954: end;
13955:
13956:
13957: procedure SIRegister_GeometryBB(CL: TPSPascalCompiler);
13958: begin
13959:   TAABB', 'record min : TAffineVector; max : TAffineVector; end
13960:   //PAABB', '^TAABB // will not work
13961:  TBSphere', 'record Center : TAffineVector; Radius : single; end
13962:  TClipRect', 'record Left : Single; Top:Single; Right:Single; Bottom : Single; end
13963:  TSpaceContains', '(scNoOverlap, scContainsFully, scContainsPartially )
13964:  Function AddBB( var c1 : THmgBoundingBox; const c2 : THmgBoundingBox) : THmgBoundingBox
13965:  Procedure AddAABB( var aabb : TAABB; const aabb1 : TAABB)
13966:  Procedure SetBB( var c : THmgBoundingBox; const v : TVector)
13967:  Procedure SetAABB( var bb : TAABB; const v : TVector)
13968:  Procedure BBTransform( var c : THmgBoundingBox; const m : TMatrix)
13969:  Procedure AABBTransform( var bb : TAABB; const m : TMatrix)
13970:  Procedure AABBScale( var bb : TAABB; const v : TAffineVector)
13971:  Function BBMinX( const c : THmgBoundingBox) : Single
13972:  Function BBMaxX( const c : THmgBoundingBox) : Single
13973:  Function BBMinY( const c : THmgBoundingBox) : Single
13974:  Function BBMaxY( const c : THmgBoundingBox) : Single
13975:  Function BBMinZ( const c : THmgBoundingBox) : Single
13976:  Function BBMaxZ( const c : THmgBoundingBox) : Single
13977:  Procedure AABBInclude( var bb : TAABB; const p : TAffineVector)
13978:  Procedure AABBFromSweep( var SweepAABB : TAABB; const Start, Dest : TVector; const Radius : Single)
13979:  Function AABBIntersection( const aabb1, aabb2 : TAABB) : TAABB
13980:  Function BBToAABB( const aBB : THmgBoundingBox) : TAABB
13981:  Function AABBToBB( const anAABB : TAABB) : THmgBoundingBox;
13982:  Function AABBToBB1( const anAABB : TAABB; const m : TMatrix) : THmgBoundingBox;
13983:  Procedure OffsetAABB( var aabb : TAABB; const delta : TAffineVector);
13984:  Procedure OffsetAABB1( var aabb : TAABB; const delta : TVector);
13985:  Function IntersectAABBs( const aabb1, aabb2 : TAABB; const m1To2, m2To1 : TMatrix) : Boolean;
13986:  Function IntersectAABBsAbsoluteXY( const aabb1, aabb2 : TAABB) : Boolean
13987:  Function IntersectAABBsAbsoluteXZ( const aabb1, aabb2 : TAABB) : Boolean
13988:  Function IntersectAABBsAbsolute( const aabb1, aabb2 : TAABB) : Boolean
13989:  Function AABBFitsInAABBAbsolute( const aabb1, aabb2 : TAABB) : Boolean
13990:  Function PointInAABB( const p : TAffineVector; const aabb : TAABB) : Boolean;
13991:  Function PointInAABB1( const p : TVector; const aabb : TAABB) : Boolean;
13992:  Function PlaneIntersectAABB( Normal : TAffineVector; d : single; aabb : TAABB) : boolean
13993:  Function TriangleIntersectAABB( const aabb : TAABB; v1, v2, v3 : TAffineVector) : boolean
13994:  Procedure ExtractAABBCorners( const AABB : TAABB; var AABBCorners : TAABBCorners)
13995:  Procedure AABBToBSphere( const AABB : TAABB; var BSphere : TBSphere)
13996:  Procedure BSphereToAABB( const BSphere : TBSphere; var AABB : TAABB);
13997:  Function BSphereToAABB1( const center : TAffineVector; radius : Single) : TAABB;
13998:  Function BSphereToAABB2( const center : TVector; radius : Single) : TAABB;
13999:  Function AABBContainsAABB( const mainAABB, testAABB : TAABB) : TSpaceContains
14000:  Function BSphereContainsAABB( const mainBSphere : TBSphere; const testAABB : TAABB) : TSpaceContains
14001:  Function BSphereContainsBSphere( const mainBSphere, testBSphere : TBSphere) : TSpaceContains
14002:  Function AABBContainsBSphere( const mainAABB : TAABB; const testBSphere : TBSphere) : TSpaceContains
14003:  Function PlaneContainsBSphere( const Location,Normal:TAffineVector;const
       testBSphere:TBSphere):TSpaceContains
14004:  Function FrustumContainsBSphere( const Frustum : TFrustum; const testBSphere : TBSphere) : TSpaceContains
14005:  Function FrustumContainsAABB( const Frustum : TFrustum; const testAABB : TAABB) : TSpaceContains
14006:  Function ClipToAABB( const v : TAffineVector; const AABB : TAABB) : TAffineVector
14007:  Function BSphereIntersectsBSphere( const mainBSphere, testBSphere : TBSphere) : boolean
14008:  Procedure IncludeInClipRect( var clipRect : TClipRect; x, y : Single)
14009:  Function AABBToClipRect(const aabb:TAABB;modelViewProjection:TMatrix;viewportSizeX,
       viewportSizeY:Int):TClipRect
14010: end;
14011:
14012: procedure SIRegister_GeometryCoordinates(CL: TPSPascalCompiler);
14013: begin
14014:  Procedure Cylindrical_Cartesian( const r, theta, z1 : single; var x, y, z : single);
14015:  Procedure Cylindrical_Cartesian1( const r, theta, z1 : double; var x, y, z : double);
14016:  Procedure Cylindrical_Cartesian2( const r,theta,z1 : single; var x, y, z : single; var ierr : integer);
14017:  Procedure Cylindrical_Cartesian3( const r,theta,z1 : double; var x, y, z : double; var ierr : integer);
14018:  Procedure Cartesian_Cylindrical( const x, y, z1 : single; var r, theta, z : single);
14019:  Procedure Cartesian_Cylindrical1( const x, y, z1 : double; var r, theta, z : double);
14020:  Procedure Spherical_Cartesian( const r, theta, phi : single; var x, y, z : single);
14021:  Procedure Spherical_Cartesian1( const r, theta, phi : double; var x, y, z : double);
14022:  Procedure Spherical_Cartesian2( const r,theta, phi : single; var x, y, z : single; var ierr : integer);
14023:  Procedure Spherical_Cartesian3( const r,theta, phi : double; var x, y, z : double; var ierr : integer);
14024:  Procedure Cartesian_Spherical( const x, y, z : single; var r, theta, phi : single);
14025:  Procedure Cartesian_Spherical1( const v : TAffineVector; var r, theta, phi : Single);
14026:  Procedure Cartesian_Spherical2( const x, y, z : double; var r, theta, phi : double);
14027:  Procedure ProlateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14028:  Procedure ProlateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14029:  Procedure ProlateSpheroidal_Cartesian2(const xi,eta,phi,a:single;var x,y,z:single;var ierr: integer);
14030:  Procedure ProlateSpheroidal_Cartesian3(const xi,eta,phi,a:double;var x,y,z:double;var ierr: integer);
14031:  Procedure OblateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14032:  Procedure OblateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14033:  Procedure OblateSpheroidal_Cartesian2(const xi,eta,phi,a:single; var x,y,z: single;var ierr:integer);
14034:  Procedure OblateSpheroidal_Cartesian3( const xi,eta,phi,a:double; var x,y,z:double;var ierr:integer);
14035:  Procedure BipolarCylindrical_Cartesian( const u, v, z1, a : single; var x, y, z : single);
14036:  Procedure BipolarCylindrical_Cartesian1(const u, v, z1, a : double; var x, y, z : double);
14037:  Procedure BipolarCylindrical_Cartesian2(const u,v,z1,a: single;var x,y,z:single; var ierr : integer);
14038:  Procedure BipolarCylindrical_Cartesian3(const u,v,z1,a: double;var x,y,z:double; var ierr : integer);
```

```
14039: end;
14040:
14041: procedure SIRegister_VectorGeometry(CL: TPSPascalCompiler);
14042: begin
14043:  'EPSILON','Single').setExtended( 1e-40);
14044:  'EPSILON2','Single').setExtended( 1e-30);  }
14045:  TRenderContextClippingInfo', 'record origin : TVector; clippingD'
14046:    +'irection : TVector; viewPortRadius : Single; farClippingDistance:Single;frustum:TFrustum; end
14047:  THmgPlane', 'TVector
14048:   TDoubleHmgPlane', 'THomogeneousDblVector
14049:  {TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
14050:    +'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
14051:    +', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )}
14052:   TSingleArray', 'array of Single
14053:   TTransformations','array [0..15] of Single)
14054:   TPackedRotationMatrix','array [0..2] of Smallint)
14055:   TVertex', 'TAffineVector
14056:   //TVectorGL', 'THomogeneousFltVector
14057:   //TMatrixGL', 'THomogeneousFltMatrix
14058:   //  TPackedRotationMatrix = array [0..2] of SmallInt;
14059:  Function glTexPointMake( const s, t : Single) : TTexPoint
14060:  Function glAffineVectorMake( const x, y, z : Single) : TAffineVector;
14061:  Function glAffineVectorMake1( const v : TVectorGL) : TAffineVector;
14062:  Procedure glSetAffineVector( var v : TAffineVector; const x, y, z : Single);
14063:  Procedure glSetVector( var v : TAffineVector; const x, y, z : Single);
14064:  Procedure glSetVector1( var v : TAffineVector; const vSrc : TVectorGL);
14065:  Procedure glSetVector2( var v : TAffineVector; const vSrc : TAffineVector);
14066:  Procedure glSetVector3( var v : TAffineDblVector; const vSrc : TAffineVector);
14067:  Procedure glSetVector4( var v : TAffineDblVector; const vSrc : TVectorGL);
14068:  Function glVectorMake( const v : TAffineVector; w : Single) : TVectorGL;
14069:  Function glVectorMake1( const x, y, z : Single; w : Single) : TVectorGL;
14070:  Function glPointMake( const x, y, z : Single) : TVectorGL;
14071:  Function glPointMake1( const v : TAffineVector) : TVectorGL;
14072:  Function glPointMake2( const v : TVectorGL) : TVectorGL;
14073:  Procedure glSetVector5( var v : TVectorGL; const x, y, z : Single; w : Single);
14074:  Procedure glSetVector6( var v : TVectorGL; const av : TAffineVector; w : Single);
14075:  Procedure glglSetVector7( var v : TVectorGL; const vSrc : TVectorGL);
14076:  Procedure glMakePoint( var v : TVectorGL; const x, y, z : Single);
14077:  Procedure glMakePoint1( var v : TVectorGL; const av : TAffineVector);
14078:  Procedure glMakePoint2( var v : TVectorGL; const av : TVectorGL);
14079:  Procedure glMakeVector( var v : TAffineVector; const x, y, z : Single);
14080:  Procedure glMakeVector1( var v : TVectorGL; const x, y, z : Single);
14081:  Procedure glMakeVector2( var v : TVectorGL; const av : TAffineVector);
14082:  Procedure glMakeVector3( var v : TVectorGL; const av : TVectorGL);
14083:  Procedure glRstVector( var v : TAffineVector);
14084:  Procedure glRstVector1( var v : TVectorGL);
14085:  Function glVectorAdd( const v1, v2 : TAffineVector) : TAffineVector;
14086:  Procedure glVectorAdd1( const v1, v2 : TAffineVector; var vr : TAffineVector);
14087:  //Procedure VectorAdd2( const v1, v2 : TAffineVector; vr : PAffineVector);
14088:  Function glVectorAdd3( const v1, v2 : TVectorGL) : TVectorGL;
14089:  Procedure glVectorAdd4( const v1, v2 : TVectorGL; var vr : TVectorGL);
14090:  Function glVectorAdd5( const v : TAffineVector; const f : Single) : TAffineVector;
14091:  Function glVectorAdd6( const v : TVectorGL; const f : Single) : TVectorGL;
14092:  Procedure glAddVector7( var v1 : TAffineVector; const v2 : TAffineVector);
14093:  Procedure glAddVector8( var v1 : TAffineVector; const v2 : TVectorGL);
14094:  Procedure glAddVector9( var v1 : TVectorGL; const v2 : TVectorGL);
14095:  Procedure glAddVector10( var v : TAffineVector; const f : Single);
14096:  Procedure glAddVector11( var v : TVectorGL; const f : Single);
14097:  //Procedure TexPointArrayAdd( const src : PTexPointArray; const delta : TTexPoint;const nb:Integer; dest
      : PTexPointArray);
14098:  //Procedure TexPointArrayScaleAndAdd( const src : PTexPointArray; const delta : TTexPoint; const nb :
      Integer; const scale : TTexPoint; dest : PTexPointArray);
14099:  //Procedure VectorArrayAdd( const src : PAffineVectorArray; const delta : TAffineVector; const nb :
      Integer; dest : PAffineVectorArray);
14100:  Function glVectorSubtract( const V1, V2 : TAffineVector) : TAffineVector;
14101:  Procedure glVectorSubtract1( const v1, v2 : TAffineVector; var result : TAffineVector);
14102:  Procedure glVectorSubtract2( const v1, v2 : TAffineVector; var result : TVectorGL);
14103:  Procedure glVectorSubtract3( const v1 : TVectorGL; v2 : TAffineVector; var result : TVectorGL);
14104:  Function glVectorSubtract4( const V1, V2 : TVectorGL) : TVectorGL;
14105:  Procedure glVectorSubtract5( const v1, v2 : TVectorGL; var result : TVectorGL);
14106:  Procedure glVectorSubtract6( const v1, v2 : TVectorGL; var result : TAffineVector);
14107:  Function glVectorSubtract7( const v1 : TAffineVector; delta : Single) : TAffineVector;
14108:  Function glVectorSubtract8( const v1 : TVectorGL; delta : Single) : TVectorGL;
14109:  Procedure glSubtractVector9( var V1 : TAffineVector; const V2 : TAffineVector);
14110:  Procedure glSubtractVector10( var V1 : TVectorGL; const V2 : TVectorGL);
14111:  Procedure glCombineVector( var vr : TAffineVector; const v : TAffineVector; var f : Single);
14112:  //Procedure CombineVector1( var vr : TAffineVector; const v : TAffineVector; pf : PFloat);
14113:  Function glTexPointCombine( const t1, t2 : TTexPoint; f1, f2 : Single) : TTexPoint
14114:  Function glVectorCombine2( const V1, V2 : TAffineVector; const F1, F2 : Single) : TAffineVector;
14115:  Function glVectorCombine33( const V1, V2, V3 : TAffineVector; const F1, F2, F3 : Single) : TAffineVector;
14116:  Procedure glVectorCombine34(const V1,V2,V3:TAffineVector; const F1,F2,F3: Single; var vr : TAffineVector);
14117:  Procedure glCombineVector5( var vr : TVectorGL; const v : TVectorGL; var f : Single);
14118:  Procedure glCombineVector6( var vr : TVectorGL; const v : TAffineVector; var f : Single);
14119:  Function glVectorCombine7( const V1, V2 : TVectorGL; const F1, F2 : Single) : TVectorGL;
14120:  Function glVectorCombine8( const V1 : TVectorGL;const V2: TAffineVector; const F1,F2:Single): TVectorGL;
14121:  Procedure glVectorCombine9(const V1:TVectorGL;const V2:TAffineVector;const F1,F2:Single;var vr:TVectorGL);
14122:  Procedure glVectorCombine10( const V1, V2 : TVectorGL; const F1, F2 : Single; var vr : TVectorGL);
14123:  Procedure glVectorCombine11( const V1, V2 : TVectorGL; const F2 : Single; var vr : TVectorGL);
14124:  Function glVectorCombine3( const V1, V2, V3 : TVectorGL; const F1, F2, F3 : Single) : TVectorGL;
```

```
14125:  Procedure glVectorCombine31( const V1, V2, V3 : TVectorGL; const F1, F2, F3:Single; var vr : TVectorGL);
14126:  Function glVectorDotProduct( const V1, V2 : TAffineVector) : Single;
14127:  Function glVectorDotProduct1( const V1, V2 : TVectorGL) : Single;
14128:  Function glVectorDotProduct2( const V1 : TVectorGL; const V2 : TAffineVector) : Single;
14129:  Function glPointProject( const p, origin, direction : TAffineVector) : Single;
14130:  Function glPointProject1( const p, origin, direction : TVectorGL) : Single;
14131:  Function glVectorCrossProduct( const V1, V2 : TAffineVector) : TAffineVector;
14132:  Function glVectorCrossProduct1( const V1, V2 : TVectorGL) : TVectorGL;
14133:  Procedure glVectorCrossProduct2( const v1, v2 : TVectorGL; var vr : TVectorGL);
14134:  Procedure glVectorCrossProduct3( const v1, v2 : TAffineVector; var vr : TVectorGL);
14135:  Procedure glVectorCrossProduct4( const v1, v2 : TVectorGL; var vr : TAffineVector);
14136:  Procedure glVectorCrossProduct5( const v1, v2 : TAffineVector; var vr : TAffineVector);
14137:  Function glLerp( const start, stop, t : Single) : Single
14138:  Function glAngleLerp( start, stop, t : Single) : Single
14139:  Function glDistanceBetweenAngles( angle1, angle2 : Single) : Single
14140:  Function glTexPointLerp( const t1, t2 : TTexPoint; t : Single) : TTexPoint;
14141:  Function glVectorLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14142:  Procedure glVectorLerp1( const v1, v2 : TAffineVector; t : Single; var vr : TAffineVector);
14143:  Function glVectorLerp2( const v1, v2 : TVectorGL; t : Single) : TVectorGL;
14144:  Procedure glVectorLerp3( const v1, v2 : TVectorGL; t : Single; var vr : TVectorGL);
14145:  Function glVectorAngleLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14146:  Function glVectorAngleCombine( const v1, v2 : TAffineVector; f : Single) : TAffineVector;
14147: // Procedure VectorArrayLerp( const src1, src2:PVectorArray; t:Single; n:Integer; dest:PVectorArray);
14148: // Procedure VectorArrayLerp1( const src1, src2 : PAffineVectorArray; t : Single; n : Integer; dest :
       PAffineVectorArray);
14149:  Function glVectorLength( const x, y : Single) : Single;
14150:  Function glVectorLength1( const x, y, z : Single) : Single;
14151:  Function glVectorLength2( const v : TAffineVector) : Single;
14152:  Function glVectorLength3( const v : TVectorGL) : Single;
14153:  Function glVectorLength4( const v : array of Single) : Single;
14154:  Function glVectorNorm( const x, y : Single) : Single;
14155:  Function glVectorNorm1( const v : TAffineVector) : Single;
14156:  Function glVectorNorm2( const v : TVectorGL) : Single;
14157:  Function glVectorNorm3( var V : array of Single) : Single;
14158:  Procedure glNormalizeVector( var v : TAffineVector);
14159:  Procedure glNormalizeVector1( var v : TVectorGL);
14160:  Function glVectorNormalize( const v : TAffineVector) : TAffineVector;
14161:  Function glVectorNormalize1( const v : TVectorGL) : TVectorGL;
14162: // Procedure NormalizeVectorArray( list : PAffineVectorArray; n : Integer);
14163:  Function glVectorAngleCosine( const V1, V2 : TAffineVector) : Single
14164:  Function glVectorNegate( const v : TAffineVector) : TAffineVector;
14165:  Function glVectorNegate1( const v : TVectorGL) : TVectorGL;
14166:  Procedure glNegateVector( var V : TAffineVector);
14167:  Procedure glNegateVector2( var V : TVectorGL);
14168:  Procedure glNegateVector3( var V : array of Single);
14169:  Procedure glScaleVector( var v : TAffineVector; factor : Single);
14170:  Procedure glScaleVector1( var v : TAffineVector; const factor : TAffineVector);
14171:  Procedure glScaleVector2( var v : TVectorGL; factor : Single);
14172:  Procedure glScaleVector3( var v : TVectorGL; const factor : TVectorGL);
14173:  Function glVectorScale( const v : TAffineVector; factor : Single) : TAffineVector;
14174:  Procedure glVectorScale1( const v : TAffineVector; factor : Single; var vr : TAffineVector);
14175:  Function glVectorScale2( const v : TVectorGL; factor : Single) : TVectorGL;
14176:  Procedure glVectorScale3( const v : TVectorGL; factor : Single; var vr : TVectorGL);
14177:  Procedure glVectorScale4( const v : TVectorGL; factor : Single; var vr : TAffineVector);
14178:  Procedure glDivideVector( var v : TVectorGL; const divider : TVectorGL);
14179:  Function glVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14180:  Function glVectorEquals1( const V1, V2 : TAffineVector) : Boolean;
14181:  Function glAffineVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14182:  Function glVectorIsNull( const v : TVectorGL) : Boolean;
14183:  Function glVectorIsNull1( const v : TAffineVector) : Boolean;
14184:  Function glVectorSpacing( const v1, v2 : TTexPoint) : Single;
14185:  Function glVectorSpacing1( const v1, v2 : TAffineVector) : Single;
14186:  Function glVectorSpacing2( const v1, v2 : TVectorGL) : Single;
14187:  Function glVectorDistance( const v1, v2 : TAffineVector) : Single;
14188:  Function glVectorDistance1( const v1, v2 : TVectorGL) : Single;
14189:  Function glVectorDistance2( const v1, v2 : TAffineVector) : Single;
14190:  Function glVectorDistance21( const v1, v2 : TVectorGL) : Single;
14191:  Function glVectorPerpendicular( const V, N : TAffineVector) : TAffineVector
14192:  Function glVectorReflect( const V, N : TAffineVector) : TAffineVector
14193:  Procedure glRotateVector( var vector : TVectorGL; const axis : TAffineVector; angle : Single);
14194:  Procedure glRotateVector1( var vector : TVectorGL; const axis : TVectorGL; angle : Single);
14195:  Procedure glRotateVectorAroundY( var v : TAffineVector; alpha : Single)
14196:  Function glVectorRotateAroundX( const v : TAffineVector; alpha : Single) : TAffineVector;
14197:  Function glVectorRotateAroundY( const v : TAffineVector; alpha : Single) : TAffineVector;
14198:  Procedure glVectorRotateAroundY1(const v : TAffineVector; alpha : Single; var vr : TAffineVector);
14199:  Function glVectorRotateAroundZ( const v : TAffineVector; alpha : Single) : TAffineVector;
14200:  Procedure glAbsVector( var v : TVectorGL);
14201:  Procedure glAbsVector1( var v : TAffineVector);
14202:  Function glVectorAbs( const v : TVectorGL) : TVectorGL;
14203:  Function glVectorAbs1( const v : TAffineVector) : TAffineVector;
14204:  Procedure glSetMatrix( var dest : THomogeneousDblMatrix; const src : TMatrixGL);
14205:  Procedure glSetMatrix1( var dest : TAffineMatrix; const src : TMatrixGL);
14206:  Procedure glSetMatrix2( var dest : TMatrixGL; const src : TAffineMatrix);
14207:  Procedure glSetMatrixRow( var dest : TMatrixGL; rowNb : Integer; const aRow : TVectorGL);
14208:  Function glCreateScaleMatrix( const v : TAffineVector) : TMatrixGL;
14209:  Function glCreateScaleMatrix1( const v : TVectorGL) : TMatrixGL;
14210:  Function glCreateTranslationMatrix( const V : TAffineVector) : TMatrixGL;
14211:  Function glCreateTranslationMatrix1( const V : TVectorGL) : TMatrixGL;
14212:  Function glCreateScaleAndTranslationMatrix( const scale, offset : TVectorGL) : TMatrixGL;
```

```
14213:  Function glCreateRotationMatrixX( const sine, cosine : Single) : TMatrixGL;
14214:  Function glCreateRotationMatrixX1( const angle : Single) : TMatrixGL;
14215:  Function glCreateRotationMatrixY( const sine, cosine : Single) : TMatrixGL;
14216:  Function glCreateRotationMatrixY1( const angle : Single) : TMatrixGL;
14217:  Function glCreateRotationMatrixZ( const sine, cosine : Single) : TMatrixGL;
14218:  Function glCreateRotationMatrixZ1( const angle : Single) : TMatrixGL;
14219:  Function glCreateRotationMatrix( const anAxis : TAffineVector; angle : Single) : TMatrixGL;
14220:  Function glCreateRotationMatrix1( const anAxis : TVectorGL; angle : Single) : TMatrixGL;
14221:  Function glCreateAffineRotationMatrix( const anAxis : TAffineVector; angle:Single):TAffineMatrix
14222:  Function glMatrixMultiply( const M1, M2 : TAffineMatrix) : TAffineMatrix;
14223:  Function glMatrixMultiply1( const M1, M2 : TMatrixGL) : TMatrixGL;
14224:  Procedure glMatrixMultiply2( const M1, M2 : TMatrixGL; var MResult : TMatrixGL);
14225:  Function glVectorTransform( const V : TVectorGL; const M : TMatrixGL) : TVectorGL;
14226:  Function glVectorTransform1( const V : TVectorGL; const M : TAffineMatrix) : TVectorGL;
14227:  Function glVectorTransform2( const V : TAffineVector; const M : TMatrixGL) : TAffineVector;
14228:  Function glVectorTransform3( const V : TAffineVector; const M : TAffineMatrix) : TAffineVector;
14229:  Function glMatrixDeterminant( const M : TAffineMatrix) : Single;
14230:  Function glMatrixDeterminant1( const M : TMatrixGL) : Single;
14231:  Procedure glAdjointMatrix( var M : TMatrixGL);
14232:  Procedure glAdjointMatrix1( var M : TAffineMatrix);
14233:  Procedure glScaleMatrix( var M : TAffineMatrix; const factor : Single);
14234:  Procedure glScaleMatrix1( var M : TMatrixGL; const factor : Single);
14235:  Procedure glTranslateMatrix( var M : TMatrixGL; const v : TAffineVector);
14236:  Procedure glTranslateMatrix1( var M : TMatrixGL; const v : TVectorGL);
14237:  Procedure glNormalizeMatrix( var M : TMatrixGL)
14238:  Procedure glTransposeMatrix( var M : TAffineMatrix);
14239:  Procedure glTransposeMatrix1( var M : TMatrixGL);
14240:  Procedure glInvertMatrix( var M : TMatrixGL);
14241:  Procedure glInvertMatrix1( var M : TAffineMatrix);
14242:  Function glAnglePreservingMatrixInvert( const mat : TMatrixGL) : TMatrixGL
14243:  Function glMatrixDecompose( const M : TMatrixGL; var Tran : TTransformations) : Boolean
14244:  Function glPlaneMake( const p1, p2, p3 : TAffineVector) : THmgPlane;
14245:  Function glPlaneMake1( const p1, p2, p3 : TVectorGL) : THmgPlane;
14246:  Function glPlaneMake2( const point, normal : TAffineVector) : THmgPlane;
14247:  Function glPlaneMake3( const point, normal : TVectorGL) : THmgPlane;
14248:  Procedure glSetPlane( var dest : TDoubleHmgPlane; const src : THmgPlane)
14249:  Procedure glNormalizePlane( var plane : THmgPlane)
14250:  Function glPlaneEvaluatePoint( const plane : THmgPlane; const point : TAffineVector) : Single;
14251:  Function glPlaneEvaluatePoint1( const plane : THmgPlane; const point : TVectorGL) : Single;
14252:  Function glCalcPlaneNormal( const p1, p2, p3 : TAffineVector) : TAffineVector;
14253:  Procedure glCalcPlaneNormal1( const p1, p2, p3 : TAffineVector; var vr : TAffineVector);
14254:  Procedure glCalcPlaneNormal2( const p1, p2, p3 : TVectorGL; var vr : TAffineVector);
14255:  Function glPointIsInHalfSpace( const point, planePoint, planeNormal : TVectorGL) : Boolean;
14256:  Function glPointIsInHalfSpace1( const point, planePoint, planeNormal : TAffineVector) : Boolean;
14257:  Function glPointPlaneDistance( const point, planePoint, planeNormal : TVectorGL) : Single;
14258:  Function glPointPlaneDistance1( const point, planePoint, planeNormal : TAffineVector) : Single;
14259:  Function glPointSegmentClosestPoint( const point, segmentStart, segmentStop:TAffineVector):TAffineVector
14260:  Function glPointSegmentDistance( const point, segmentStart, segmentStop : TAffineVector) : single
14261:  Function glPointLineClosestPoint( const point, linePoint, lineDirection : TAffineVector) : TAffineVector
14262:  Function glPointLineDistance( const point, linePoint, lineDirection : TAffineVector) : Single
14263:  Procedure SglegmentSegmentClosestPoint( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector; var
      Segment0Closest, Segment1Closest : TAffineVector)
14264:  Function glSegmentSegmentDistance( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector) : single
14265:  TEulerOrder', '( eulXYZ, eulXZY, eulYXZ, eulYZX, eulZXY, eulZYX)
14266:  Function glQuaternionMake( const Imag : array of Single; Real : Single) : TQuaternion
14267:  Function glQuaternionConjugate( const Q : TQuaternion) : TQuaternion
14268:  Function glQuaternionMagnitude( const Q : TQuaternion) : Single
14269:  Procedure glNormalizeQuaternion( var Q : TQuaternion)
14270:  Function glQuaternionFromPoints( const V1, V2 : TAffineVector) : TQuaternion
14271:  Procedure glQuaternionToPoints( const Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
14272:  Function glQuaternionFromMatrix( const mat : TMatrixGL) : TQuaternion
14273:  Function glQuaternionToMatrix( quat : TQuaternion) : TMatrixGL
14274:  Function glQuaternionToAffineMatrix( quat : TQuaternion) : TAffineMatrix
14275:  Function glQuaternionFromAngleAxis( const angle : Single; const axis : TAffineVector) : TQuaternion
14276:  Function glQuaternionFromRollPitchYaw( const r, p, y : Single) : TQuaternion
14277:  Function glQuaternionFromEuler( const x, y, z : Single; eulerOrder : TEulerOrder) : TQuaternion
14278:  Function glQuaternionMultiply( const qL, qR : TQuaternion) : TQuaternion
14279:  Function glQuaternionSlerp( const QStart, QEnd : TQuaternion; Spin : Integer; t : Single) : TQuaternion;
14280:  Function glQuaternionSlerp1( const source, dest : TQuaternion; const t : Single) : TQuaternion;
14281:  Function glLnXP1( X : Extended) : Extended
14282:  Function glLog10( X : Extended) : Extended
14283:  Function glLog2( X : Extended) : Extended;
14284:  Function glLog21( X : Single) : Single;
14285:  Function glLogN( Base, X : Extended) : Extended
14286:  Function glIntPower( Base : Extended; Exponent : Integer) : Extended
14287:  Function glPower( const Base, Exponent : Single) : Single;
14288:  Function glPower1( Base : Single; Exponent : Integer) : Single;
14289:  Function glDegToRad( const Degrees : Extended) : Extended;
14290:  Function glDegToRad1( const Degrees : Single) : Single;
14291:  Function glRadToDeg( const Radians : Extended) : Extended;
14292:  Function glRadToDeg1( const Radians : Single) : Single;
14293:  Function glNormalizeAngle( angle : Single) : Single
14294:  Function glNormalizeDegAngle( angle : Single) : Single
14295:  Procedure glSinCos( const Theta : Extended; var Sin, Cos : Extended);
14296:  Procedure glSinCos1( const Theta : Double; var Sin, Cos : Double);
14297:  Procedure glSinCos( const Theta : Single; var Sin, Cos : Single);
14298:  Procedure glSinCos1( const theta, radius : Double; var Sin, Cos : Extended);
14299:  Procedure glSinCos2( const theta, radius : Double; var Sin, Cos : Double);
14300:  Procedure glSinCos3( const theta, radius : Single; var Sin, Cos : Single);
```

```
14301:  Procedure glPrepareSinCosCache( var s, c : array of Single; startAngle, stopAngle : Single)
14302:  Function glArcCos( const X : Extended) : Extended;
14303:  Function glArcCos1( const x : Single) : Single;
14304:  Function glArcSin( const X : Extended) : Extended;
14305:  Function glArcSin1( const X : Single) : Single;
14306:  Function glArcTan21( const Y, X : Extended) : Extended;
14307:  Function glArcTan21( const Y, X : Single) : Single;
14308:  Function glFastArcTan2( y, x : Single) : Single
14309:  Function glTan( const X : Extended) : Extended;
14310:  Function glTan1( const X : Single) : Single;
14311:  Function glCoTan( const X : Extended) : Extended;
14312:  Function glCoTan1( const X : Single) : Single;
14313:  Function glSinh( const x : Single) : Single;
14314:  Function glSinh1( const x : Double) : Double;
14315:  Function glCosh( const x : Single) : Single;
14316:  Function glCosh1( const x : Double) : Double;
14317:  Function glRSqrt( v : Single) : Single
14318:  Function glRLength( x, y : Single) : Single
14319:  Function glISqrt( i : Integer) : Integer
14320:  Function glILength( x, y : Integer) : Integer;
14321:  Function glILength1( x, y, z : Integer) : Integer;
14322:  Procedure glRegisterBasedExp
14323:  Procedure glRandomPointOnSphere( var p : TAffineVector)
14324:  Function glRoundInt( v : Single) : Single;
14325:  Function glRoundInt1( v : Extended) : Extended;
14326:  Function glTrunc( v : Single) : Integer;
14327:  Function glTrunc64( v : Extended) : Int64;
14328:  Function glInt( v : Single) : Single;
14329:  Function glInt1( v : Extended) : Extended;
14330:  Function glFrac( v : Single) : Single;
14331:  Function glFrac1( v : Extended) : Extended;
14332:  Function glRound( v : Single) : Integer;
14333:  Function glRound64( v : Single) : Int64;
14334:  Function glRound641( v : Extended) : Int64;
14335:  Function glTrunc( X : Extended) : Int64
14336:  Function glRound( X : Extended) : Int64
14337:  Function glFrac( X : Extended) : Extended
14338:  Function glCeil( v : Single) : Integer;
14339:  Function glCeil64( v : Extended) : Int64;
14340:  Function glFloor( v : Single) : Integer;
14341:  Function glFloor64( v : Extended) : Int64;
14342:  Function glScaleAndRound( i : Integer; var s : Single) : Integer
14343:  Function glSign( x : Single) : Integer
14344:  Function glIsInRange( const x, a, b : Single) : Boolean;
14345:  Function glIsInRange1( const x, a, b : Double) : Boolean;
14346:  Function glIsInCube( const p, d : TAffineVector) : Boolean;
14347:  Function glIsInCube1( const p, d : TVectorGL) : Boolean;
14348:  //Function MinFloat( values : PSingleArray; nbItems : Integer) : Single;
14349:  //Function MinFloat1( values : PDoubleArray; nbItems : Integer) : Double;
14350:  //Function MinFloat2( values : PExtendedArray; nbItems : Integer) : Extended;
14351:  Function glMinFloat3( const v1, v2 : Single) : Single;
14352:  Function glMinFloat4( const v : array of Single) : Single;
14353:  Function glMinFloat5( const v1, v2 : Double) : Double;
14354:  Function glMinFloat6( const v1, v2 : Extended) : Extended;
14355:  Function glMinFloat7( const v1, v2, v3 : Single) : Single;
14356:  Function glMinFloat8( const v1, v2, v3 : Double) : Double;
14357:  Function glMinFloat9( const v1, v2, v3 : Extended) : Extended;
14358:  //Function MaxFloat10( values : PSingleArray; nbItems : Integer) : Single;
14359:  //Function MaxFloat( values : PDoubleArray; nbItems : Integer) : Double;
14360:  //Function MaxFloat1( values : PExtendedArray; nbItems : Integer) : Extended;
14361:  Function glMaxFloat2( const v : array of Single) : Single;
14362:  Function glMaxFloat3( const v1, v2 : Single) : Single;
14363:  Function glMaxFloat4( const v1, v2 : Double) : Double;
14364:  Function glMaxFloat5( const v1, v2 : Extended) : Extended;
14365:  Function glMaxFloat6( const v1, v2, v3 : Single) : Single;
14366:  Function glMaxFloat7( const v1, v2, v3 : Double) : Double;
14367:  Function glMaxFloat8( const v1, v2, v3 : Extended) : Extended;
14368:  Function glMinInteger9( const v1, v2 : Integer) : Integer;
14369:  Function glMinInteger( const v1, v2 : Cardinal) : Cardinal;
14370:  Function glMaxInteger( const v1, v2 : Integer) : Integer;
14371:  Function glMaxInteger1( const v1, v2 : Cardinal) : Cardinal;
14372:  Function glTriangleArea( const p1, p2, p3 : TAffineVector) : Single;
14373:  //Function PolygonArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14374:  Function glTriangleSignedArea( const p1, p2, p3 : TAffineVector) : Single;
14375:  //Function PolygonSignedArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14376:  //Procedure ScaleFloatArray( values : PSingleArray; nb : Integer; var factor : Single);
14377:  Procedure glScaleFloatArray( var values : TSingleArray; factor : Single);
14378:  //Procedure OffsetFloatArray( values : PSingleArray; nb : Integer; var delta : Single);
14379:  Procedure glOffsetFloatArray1( var values : array of Single; delta : Single);
14380:  //Procedure OffsetFloatArray2( valuesDest, valuesDelta : PSingleArray; nb : Integer);
14381:  Function glMaxXYZComponent( const v : TVectorGL) : Single;
14382:  Function glMaxXYZComponent1( const v : TAffineVector) : single;
14383:  Function glMinXYZComponent( const v : TVectorGL) : Single;
14384:  Function glMinXYZComponent1( const v : TAffineVector) : single;
14385:  Function glMaxAbsXYZComponent( v : TVectorGL) : Single
14386:  Function glMinAbsXYZComponent( v : TVectorGL) : Single
14387:  Procedure glMaxVector( var v : TVectorGL; const v1 : TVectorGL);
14388:  Procedure glMaxVector1( var v : TAffineVector; const v1 : TAffineVector);
14389:  Procedure glMinVector( var v : TVectorGL; const v1 : TVectorGL);
```

```
14390:  Procedure glMinVector1( var v : TAffineVector; const v1 : TAffineVector);
14391:  Procedure glSortArrayAscending( var a : array of Extended)
14392:  Function glClampValue( const aValue, aMin, aMax : Single) : Single;
14393:  Function glClampValue1( const aValue, aMin : Single) : Single;
14394:  Function glGeometryOptimizationMode : String
14395:  Procedure glBeginFPUOnlySection
14396:  Procedure glEndFPUOnlySection
14397:  Function glConvertRotation( const Angles : TAffineVector) : TVectorGL
14398:  Function glMakeAffineDblVector( var v : array of Double) : TAffineDblVector
14399:  Function glMakeDblVector( var v : array of Double) : THomogeneousDblVector
14400:  Function glVectorAffineDblToFlt( const v : TAffineDblVector) : TAffineVector
14401:  Function glVectorDblToFlt( const v : THomogeneousDblVector) : THomogeneousVector
14402:  Function glVectorAffineFltToDbl( const v : TAffineVector) : TAffineDblVector
14403:  Function glVectorFltToDbl( const v : TVectorGL) : THomogeneousDblVector
14404:  Function glPointInPolygon( var xp, yp : array of Single; x, y : Single) : Boolean
14405:  Procedure glDivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
14406:  Function glTurn( const Matrix : TMatrixGL; angle : Single) : TMatrixGL;
14407:  Function glTurn1( const Matrix : TMatrixGL; const MasterUp:TAffineVector;Angle:Single):TMatrixGL;
14408:  Function glPitch( const Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
14409:  Function glPitch1( const Matrix:TMatrixGL;const MasterRight:TAffineVector;Angle:Single):TMatrixGL;
14410:  Function glRoll( const Matrix: TMatrixGL; Angle : Single) : TMatrixGL;
14411:  Function glRoll1(const Matrix: TMatrixGL;const MasterDirection:TAffineVector;Angle: Single): TMatrixGL;
14412:  Function glRayCastMinDistToPoint( const rayStart, rayVector: TVectorGL;const point:TVectorGL):Single
14413:  Function glRayCastIntersectsSphere( const rayStart, rayVector : TVectorGL; const
       sphereCenter:TVectorGL;const sphereRadius : Single) : Boolean;
14414:  Function glRayCastSphereIntersect( const rayStart, rayVector : TVectorGL; const sphereCenter:TVectorGL;
       const sphereRadius : Single; var i1, i2 : TVectorGL) : Integer;
14415:  Function glSphereVisibleRadius( distance, radius : Single) : Single
14416:  Function glExtractFrustumFromModelViewProjection( const modelViewProj : TMatrixGL) : TFrustum
14417:  Function glIsVolumeClipped( const objPos : TVectorGL; const objRadius : Single; const rcci :
       TRenderContextClippingInfo) : Boolean;
14418:  Function glIsVolumeClipped1( const objPos : TAffineVector; const objRadius : Single; const rcci :
       TRenderContextClippingInfo) : Boolean;
14419:  Function glIsVolumeClipped2(const min,max:TAffineVector;const rcci : TRenderContextClippingInfo) : Bool;
14420:  Function glIsVolumeClipped3(const objPos:TAffineVector;const objRadius:Single;const
       Frustum:TFrustum):Bool;
14421:  Function glMakeParallelProjectionMatrix( const plane : THmgPlane; const dir : TVectorGL) : TMatrixGL
14422:  Function glMakeShadowMatrix( const planePoint, planeNormal, lightPos : TVectorGL) : TMatrixGL
14423:  Function glMakeReflectionMatrix( const planePoint, planeNormal : TAffineVector) : TMatrixGL
14424:  Function glPackRotationMatrix( const mat : TMatrixGL) : TPackedRotationMatrix
14425:  Function glUnPackRotationMatrix( const packedMatrix : TPackedRotationMatrix) : TMatrixGL
14426:  'cPI','Single').setExtended( 3.141592654);
14427:  'cPIdiv180','Single').setExtended( 0.017453292);
14428:  'c180divPI','Single').setExtended( 57.29577951);
14429:  'c2PI','Single').setExtended( 6.283185307);
14430:  'cPIdiv2','Single').setExtended( 1.570796326);
14431:  'cPIdiv4','Single').setExtended( 0.785398163);
14432:  'c3PIdiv4','Single').setExtended( 2.35619449);
14433:  'cInv2PI','Single').setExtended( 1 / 6.283185307);
14434:  'cInv360','Single').setExtended( 1 / 360);
14435:  'c180','Single').setExtended( 180);
14436:  'c360','Single').setExtended( 360);
14437:  'cOneHalf','Single').setExtended( 0.5);
14438:  'cLn10','Single').setExtended( 2.302585093);
14439:  {'MinSingle','Extended').setExtended( 1.5e-45);
14440:  'MaxSingle','Extended').setExtended( 3.4e+38);
14441:  'MinDouble','Extended').setExtended( 5.0e-324);
14442:  'MaxDouble','Extended').setExtended( 1.7e+308);
14443:  'MinExtended','Extended').setExtended( 3.4e-4932);
14444:  'MaxExtended','Extended').setExtended( 1.1e+4932);
14445:  'MinComp','Extended').setExtended( - 9.223372036854775807e+18);
14446:  'MaxComp','Extended').setExtended( 9.223372036854775807e+18);}
14447: end;
14448:
14449: procedure SIRegister_GLVectorFileObjects(CL: TPSPascalCompiler);
14450: begin
14451:   CL.AddClassN(CL.FindClass('TOBJECT'),'TMeshObjectList
14452:   CL.AddClassN(CL.FindClass('TOBJECT'),'TFaceGroups
14453:   TMeshAutoCentering', '( macCenterX, macCenterY, macCenterZ, macUseBarycenter )
14454:   TMeshAutoCenterings', 'set of TMeshAutoCentering
14455:   TMeshObjectMode', '( momTriangles, momTriangleStrip, momFaceGroups )
14456:   SIRegister_TBaseMeshObject(CL);
14457:   CL.AddClassN(CL.FindClass('TOBJECT'),'TSkeletonFrameList
14458:   TSkeletonFrameTransform', '( sftRotation, sftQuaternion )
14459:   SIRegister_TSkeletonFrame(CL);
14460:   SIRegister_TSkeletonFrameList(CL);
14461:   CL.AddClassN(CL.FindClass('TOBJECT'),'TSkeleton
14462:   CL.AddClassN(CL.FindClass('TOBJECT'),'TSkeletonBone
14463:   SIRegister_TSkeletonBoneList(CL);
14464:   SIRegister_TSkeletonRootBoneList(CL);
14465:   SIRegister_TSkeletonBone(CL);
14466:   CL.AddClassN(CL.FindClass('TOBJECT'),'TSkeletonColliderList
14467:   SIRegister_TSkeletonCollider(CL);
14468:   SIRegister_TSkeletonColliderList(CL);
14469:   CL.AddClassN(CL.FindClass('TOBJECT'),'TGLBaseMesh
14470:   TBlendedLerpInfo', 'record frameIndex1 : Integer; frameIndex2 : '
14471:    +'Integer; lerpFactor : Single; weight : Single; externalPositions : TAffine'
14472:    +'VectorList; externalRotations : TAffineVectorList; externalQuaternions : T'
14473:    +'QuaternionList; end
```

```
14474:   SIRegister_TSkeleton(CL);
14475:   TMeshObjectRenderingOption', '( moroGroupByMaterial )
14476:   TMeshObjectRenderingOptions', 'set of TMeshObjectRenderingOption
14477:   SIRegister_TMeshObject(CL);
14478:   SIRegister_TMeshObjectList(CL);
14479:   //TMeshObjectListClass', 'class of TMeshObjectList
14480:   CL.AddClassN(CL.FindClass('TOBJECT'),'TMeshMorphTargetList
14481:   SIRegister_TMeshMorphTarget(CL);
14482:   SIRegister_TMeshMorphTargetList(CL);
14483:   SIRegister_TMorphableMeshObject(CL);
14484:   TVertexBoneWeight', 'record BoneID : Integer; Weight : Single; end
14485:   //PVertexBoneWeightArray', '^TVertexBoneWeightArray // will not wo'rk
14486:   //PVerticesBoneWeights', '^TVerticesBoneWeights // will not work
14487:   TVertexBoneWeightDynArray', 'array of TVertexBoneWeight
14488:   SIRegister_TSkeletonMeshObject(CL);
14489:   SIRegister_TFaceGroup(CL);
14490:   TFaceGroupMeshMode', '( fgmmTriangles, fgmmTriangleStrip, fgmmFl'
14491:    +'atTriangles, fgmmTriangleFan, fgmmQuads )
14492:   SIRegister_TFGVertexIndexList(CL);
14493:   SIRegister_TFGVertexNormalTexIndexList(CL);
14494:   SIRegister_TFGIndexTexCoordList(CL);
14495:   SIRegister_TFaceGroups(CL);
14496:   TMeshNormalsOrientation', '( mnoDefault, mnoInvert )
14497:   SIRegister_TVectorFile(CL);
14498:   //TVectorFileClass', 'class of TVectorFile
14499:   SIRegister_TGLGLSMVectorFile(CL);
14500:   SIRegister_TGLBaseMesh(CL);
14501:   SIRegister_TGLFreeForm(CL);
14502:   TGLActorOption', '( aoSkeletonNormalizeNormals )
14503:   TGLActorOptions', 'set of TGLActorOption
14504:   'cDefaultGLActorOptions','LongInt').Value.ts32:= ord(aoSkeletonNormalizeNormals);
14505:   CL.AddClassN(CL.FindClass('TOBJECT'),'TGLActor
14506:   TActorAnimationReference', '( aarMorph, aarSkeleton, aarNone )
14507:   SIRegister_TActorAnimation(CL);
14508:   TActorAnimationName', 'String
14509:   SIRegister_TActorAnimations(CL);
14510:   SIRegister_TGLBaseAnimationControler(CL);
14511:   SIRegister_TGLAnimationControler(CL);
14512:   TActorFrameInterpolation', '( afpNone, afpLinear )
14513:   TActorAnimationMode', '( aamNone, aamPlayOnce, aamLoop, aamBounc'
14514:    +'eForward, aamBounceBackward, aamLoopBackward, aamExternal )
14515:   SIRegister_TGLActor(CL);
14516:   SIRegister_TVectorFileFormat(CL);
14517:   SIRegister_TVectorFileFormatsList(CL);
14518:   CL.AddClassN(CL.FindClass('TOBJECT'),'EInvalidVectorFile
14519:  Function GetVectorFileFormats : TVectorFileFormatsList
14520:  Function VectorFileFormatsFilter : String
14521:  Function VectorFileFormatsSaveFilter : String
14522:  Function VectorFileFormatExtensionByIndex( index : Integer) : String
14523:  Procedure RegisterVectorFileFormat( const aExtension, aDescription : String; aClass : TVectorFileClass)
14524:  Procedure UnregisterVectorFileClass( aClass : TVectorFileClass)
14525: end;
14526:
14527: procedure SIRegister_AxCtrls(CL: TPSPascalCompiler);
14528: begin
14529:  'Class_DColorPropPage','TGUID').SetString( '{5CFF5D59-5946-11D0-BDEF-00A024D1875C}
14530:  'Class_DFontPropPage','TGUID').SetString( '{5CFF5D5B-5946-11D0-BDEF-00A024D1875C}
14531:  'Class_DPicturePropPage','TGUID').SetString( '{5CFF5D5A-5946-11D0-BDEF-00A024D1875C}
14532:  'Class_DStringPropPage','TGUID').SetString( '{F42D677E-754B-11D0-BDFB-00A024D1875C}
14533:   SIRegister_TOleStream(CL);
14534:   CL.AddClassN(CL.FindClass('TOBJECT'),'TConnectionPoints
14535:   TConnectionKind', '( ckSingle, ckMulti )
14536:   SIRegister_TConnectionPoint(CL);
14537:   SIRegister_TConnectionPoints(CL);
14538:   TDefinePropertyPage', 'Procedure ( const GUID : TGUID)
14539:   CL.AddClassN(CL.FindClass('TOBJECT'),'TActiveXControlFactory
14540:   SIRegister_TActiveXControl(CL);
14541:   //TActiveXControlClass', 'class of TActiveXControl
14542:   SIRegister_TActiveXControlFactory(CL);
14543:   SIRegister_TActiveFormControl(CL);
14544:   SIRegister_TActiveForm(CL);
14545:   //TActiveFormClass', 'class of TActiveForm
14546:   SIRegister_TActiveFormFactory(CL);
14547:   CL.AddClassN(CL.FindClass('TOBJECT'),'TPropertyPageImpl
14548:   SIRegister_TPropertyPage(CL);
14549:   //TPropertyPageClass', 'class of TPropertyPage
14550:   SIRegister_TPropertyPageImpl(CL);
14551:   SIRegister_TActiveXPropertyPage(CL);
14552:   SIRegister_TActiveXPropertyPageFactory(CL);
14553:   SIRegister_TCustomAdapter(CL);
14554:   SIRegister_TAdapterNotifier(CL);
14555:   SIRegister_IFontAccess(CL);
14556:   SIRegister_TFontAdapter(CL);
14557:   SIRegister_IPictureAccess(CL);
14558:   SIRegister_TPictureAdapter(CL);
14559:   SIRegister_TOleGraphic(CL);
14560:   SIRegister_TStringsAdapter(CL);
14561:   SIRegister_TReflectorWindow(CL);
14562:  Procedure EnumDispatchProperties(Dispatch:IDispatch;PropType:TGUID;VTCode:Int;PropList:TStrings);
```

```
14563:  Procedure GetOleFont( Font : TFont; var OleFont : IFontDisp)
14564:  Procedure SetOleFont( Font : TFont; OleFont : IFontDisp)
14565:  Procedure GetOlePicture( Picture : TPicture; var OlePicture : IPictureDisp)
14566:  Procedure SetOlePicture( Picture : TPicture; OlePicture : IPictureDisp)
14567:  Procedure GetOleStrings( Strings : TStrings; var OleStrings : IStrings)
14568:  Procedure SetOleStrings( Strings : TStrings; OleStrings : IStrings)
14569:  Function ParkingWindow : HWND
14570: end;
14571:
14572:
14573: Functions_max hex in the box maXbox
14574: functionslist.txt
14575: FunctionsList1 3.9.9.86
14576:
14577: ********************************************************************************
14578: Procedure
14579: PROCEDURE SIZE 6792 6310 5971 4438 3797 3600 3385 3296 2883 2255 (2065) (1854)
14580: Procedure **************************Now the Procedure list******************
14581: Procedure ( ACol, ARow : Integer; Items : TStrings)
14582: Procedure ( Agg : TAggregate)
14583: Procedure ( ASender : TComponent; const AReplyStatus : TReplyStatus)
14584: Procedure ( ASender : TComponent; const AString : String; var AMsg : TIdMessage)
14585: Procedure ( ASender : TComponent; var AMsg : TIdMessage)
14586: Procedure ( ASender : TObject; const ABytes : Integer)
14587: Procedure ( ASender : TObject; VStream : TStream)
14588: Procedure ( AThread : TIdThread)
14589: Procedure ( AWebModule : TComponent)
14590: Procedure ( Column : TColumn)
14591: Procedure ( const AUsername : String; const APassword : String; AAuthenticationResult : Boolean)
14592: Procedure ( const iStart : integer; const sText : string)
14593: Procedure ( Control : TCustomTabControl; TabIndex : Integer; const Rect : TRect; Active : Boolean)
14594: Procedure ( Database : TDatabase; LoginParams : TStrings)
14595: Procedure (DataSet:TCustomClientDataSet;E:EReconcileError;UpdateKind:TUpdateKind;var Action:
       TReconcileAction)
14596: Procedure ( DATASET : TDATASET)
14597: Procedure ( DataSet:TDataSet; E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
14598: Procedure ( DATASET : TDATASET; E : TObject; var ACTION : TDATAACTION)
14599: Procedure ( DataSet : TDataSet; UpdateKind : TUpdateKind; var UpdateAction : TUpdateAction)
14600: Procedure ( DATASET : TDATASET; var ACCEPT : BOOLEAN)
14601: Procedure ( DBCtrlGrid : TDBCtrlGrid; Index : Integer)
14602: Procedure ( Done : Integer)
14603: Procedure ( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
14604: Procedure (HeaderControl:TCustomHeaderControl;Section:THeaderSection; const Rect:TRect;Pressed:Boolean)
14605: Procedure
       (HeaderControl:TCustomHeaderControl;Section:THeaderSection;Width:Integer;State:TSectionTrackState)
14606: Procedure ( HeaderControl : THeaderControl; Section : THeaderSection)
14607: Procedure (HeaderControl:THeaderControl;Section: THeaderSection; const Rect:TRect; Pressed : Boolean)
14608: Procedure (HeaderControl:THeaderControl;Section:THeaderSection;Width:Integer;State : TSectionTrackState)
14609: Procedure (Sender: TCustomListView;const ARect: TRect;Stage:TCustomDrawStage;var DefaultDraw: Boolean)
14610: Procedure ( Sender : TCustomListView; const ARect : TRect; var DefaultDraw : Boolean)
14611: Procedure ( Sender : TCustomListView; Item : TListItem; Rect : TRect; State : TOwnerDrawState)
14612: Procedure ( Sender : TCustomTreeView; const ARect : TRect; var DefaultDraw : Boolean)
14613: Procedure ( SENDER : TFIELD; const TEXT : String)
14614: Procedure ( SENDER : TFIELD; var TEXT : STRING; DISPLAYTEXT : BOOLEAN)
14615: Procedure ( Sender : TIdTelnet; const Buffer : String)
14616: Procedure ( Sender : TIdTelnet; Status : TIdTelnetCommand)
14617: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; SELECTED : BOOLEAN)
14618: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
14619: Procedure ( SENDER : TOBJECT; ACANVAS : TCANVAS; var WIDTH, HEIGHT : INTEGER)
14620: Procedure ( Sender : TObject; ACol, ARow : Longint; const Value : string)
14621: Procedure ( Sender : TObject; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
14622: Procedure ( Sender : TObject; ACol, ARow : Longint; var CanSelect : Boolean)
14623: Procedure ( Sender : TObject; ACol, ARow : Longint; var Value : string)
14624: Procedure ( Sender : TObject; Button : TMPBtnType)
14625: Procedure ( Sender : TObject; Button : TMPBtnType; var DoDefault : Boolean)
14626: Procedure ( Sender : TObject; Button : TUDBtnType)
14627: Procedure ( Sender : TObject; Canvas : TCanvas; PageRect : TRect; var DoneDrawing : Boolean)
14628: Procedure ( Sender: TObject; ClientSocket: TServerClientWinSocket; var SocketThread : TServerClientThread)
14629: Procedure ( Sender : TObject; Column : TListColumn)
14630: Procedure ( Sender : TObject; Column : TListColumn; Point : TPoint)
14631: Procedure ( Sender : TObject; Connecting : Boolean)
14632: Procedure (Sender:TObject;const PapSize:SmallInt;const Orient:TPrinterOrient;const PageTy:TPageTy;var
       DoneDrawing:Bool
14633: Procedure (Sender:TObject; const Rect : TRect; DataCol : Integer; Column: TColumn; State : TGridDrawState)
14634: Procedure ( Sender : TObject; const Rect : TRect; Field : TField; State : TGridDrawState)
14635: Procedure (Sender: TObject; const UserString:string;var DateAndTime:TDateTime;var AllowChange:Boolean)
14636: Procedure ( Sender : TObject; E : Exception; var Handled : Boolean)
14637: Procedure ( Sender : TObject; FromIndex, ToIndex : Longint)
14638: Procedure ( Sender : TObject; FromSection, ToSection : THeaderSection; var AllowDrag : Boolean)
14639: Procedure ( Sender : TObject; Index : LongInt)
14640: Procedure ( Sender : TObject; Item : TListItem)
14641: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange)
14642: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange; var AllowChange : Boolean)
14643: Procedure ( Sender : TObject; Item : TListItem; Selected : Boolean)
14644: Procedure ( Sender : TObject; Item : TListItem; var AllowEdit : Boolean)
14645: Procedure ( Sender : TObject; Item : TListItem; var S : string)
14646: Procedure ( Sender : TObject; Item1, Item2 : TListItem; Data : Integer; var Compare : Integer)
14647: Procedure ( Sender : TObject; ModalResult : TModalResult; var CanClose : Boolean)
14648: Procedure ( Sender : TObject; Month : LongWord; var MonthBoldInfo : LongWord)
```

```
14649: Procedure ( Sender : TObject; NewTab : Integer; var AllowChange : Boolean)
14650: Procedure ( Sender : TObject; Node : TTreeNode)
14651: Procedure ( Sender : TObject; Node : TTreeNode; var AllowChange : Boolean)
14652: Procedure ( Sender : TObject; Node : TTreeNode; var AllowCollapse : Boolean)
14653: Procedure ( Sender : TObject; Node : TTreeNode; var AllowEdit : Boolean)
14654: Procedure ( Sender : TObject; Node : TTreeNode; var AllowExpansion : Boolean)
14655: Procedure ( Sender : TObject; Node : TTreeNode; var S : string)
14656: Procedure ( Sender : TObject; Node1, Node2 : TTreeNode; Data : Integer; var Compare : Integer)
14657: Procedure ( Sender : TObject; NumObjects, NumChars : Integer; var SaveClipboard : Boolean)
14658: Procedure ( Sender : TObject; Rect : TRect)
14659: Procedure ( Sender : TObject; Request : TWebRequest; Response : TWebResponse; var Handled : Boolean)
14660: Procedure (Sender:TObject;Shift:TShiftState;X,Y:Integer;Orient:TPageScrollerOrientation;var Delta:Int
14661: Procedure ( Sender : TObject; Socket : TCustomWinSocket)
14662: Procedure (Sender:TObject; Socket:TCustomWinSocket;ErrorEvent: TErrorEvent; var ErrorCode : Integer)
14663: Procedure ( Sender : TObject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
14664: Procedure ( Sender : TObject; Socket : TSocket; var ClientSocket : TServerClientWinSocket)
14665: Procedure ( SENDER : TOBJECT; SOURCE : TMENUITEM; REBUILD : BOOLEAN)
14666: Procedure ( Sender : TObject; StartPos, EndPos : Integer; var AllowChange : Boolean)
14667: Procedure ( Sender : TObject; TabCanvas : TCanvas; R : TRect; Index : Integer; Selected : Boolean)
14668: Procedure ( Sender : TObject; TabIndex : Integer; var ImageIndex : Integer)
14669: Procedure ( Sender : TObject; Thread : TServerClientThread)
14670: Procedure ( Sender : TObject; TickCount : Cardinal; var Reset : Boolean)
14671: Procedure ( Sender : TObject; Username, Password : string)
14672: Procedure ( Sender : TObject; var AllowChange : Boolean)
14673: Procedure ( Sender : TObject; var AllowChange : Boolean; NewValue : SmallInt; Direction:TUpDownDirection)
14674: Procedure ( Sender : TObject; var Caption : string; var Alignment : THTMLCaptionAlignment)
14675: Procedure ( Sender : TObject; var Continue : Boolean)
14676: Procedure (Sender:TObject;var dest:string;var NumRedirect:Int;var Handled:bool; var VMethod:TIdHTTPMethod)
14677: Procedure ( Sender : TObject; var Username : string)
14678: Procedure ( Sender : TObject; Wnd : HWND)
14679: Procedure ( Sender : TToolbar; Button : TToolButton)
14680: Procedure ( Sender : TToolBar; const ARect : TRect; Stage : TCustomDrawStage; var DefaultDraw : Boolean)
14681: Procedure ( Sender : TToolBar; const ARect : TRect; var DefaultDraw : Boolean)
14682: Procedure ( Sender : TToolbar; Index : Integer; var Allow : Boolean)
14683: Procedure ( Sender : TToolbar; Index : Integer; var Button : TToolButton)
14684: Procedure ( StatusBar : TCustomStatusBar; Panel : TStatusPanel; const Rect : TRect)
14685: Procedure ( StatusBar : TStatusBar; Panel : TStatusPanel; const Rect : TRect)
14686: Procedure (var FieldNames:TWideStrings;SQL:WideString; var BindAllFields:Boolean;var TableName: WideString)
14687: Procedure ( var FieldNames : TWideStrings; SQL : WideString; var TableName : WideString)
14688: procedure (Sender: TObject)
14689: procedure (Sender: TObject; var Done: Boolean)
14690: procedure (Sender: TObject; var Key: Word; Shift: TShiftState);
14691: procedure _T(Name: tbtString; v: Variant);
14692: Procedure AbandonSignalHandler( RtlSigNum : Integer)
14693: Procedure Abort
14694: Procedure About1Click( Sender : TObject)
14695: Procedure Accept( Socket : TSocket)
14696: Procedure AESSymetricExecute(const plaintext, ciphertext, password: string)
14697: Procedure AESEncryptFile(const plaintext, ciphertext, password: string)
14698: Procedure AESDecryptFile(const replaintext, ciphertext, password: string)
14699: Procedure AESEncryptString(const plaintext: string; var ciphertext: string; password: string)
14700: Procedure AESDecryptString(var plaintext: string; const ciphertext: string; password: string)
14701: Procedure Add( Addend1, Addend2 : TMyBigInt)
14702: Procedure ADD( const AKEY, AVALUE : VARIANT)
14703: Procedure Add( const Key : string; Value : Integer)
14704: Procedure ADD( const NAME, FIELDS : String; OPTIONS : TINDEXOPTIONS)
14705: Procedure ADD( FIELD : TFIELD)
14706: Procedure ADD( ITEM : TMENUITEM)
14707: Procedure ADD( POPUP : TPOPUPMENU)
14708: Procedure AddCharacters( xCharacters : TCharSet)
14709: Procedure AddDriver( const Name : string; List : TStrings)
14710: Procedure AddImages( Value : TCustomImageList)
14711: Procedure AddIndex( const Name, Fields : string; Options : TIndexOptions; const DescFields : string)
14712: Procedure AddLambdaTransitionTo( oState : TniRegularExpressionState)
14713: Procedure AddLoader( Loader : TBitmapLoader)
14714: Procedure ADDPARAM( VALUE : TPARAM)
14715: Procedure AddPassword( const Password : string)
14716: Procedure AddStandardAlias( const Name, Path, DefaultDriver : string)
14717: Procedure AddState( oState : TniRegularExpressionState)
14718: Procedure AddStrings( Strings : TStrings);
14719: procedure AddStrings(Strings: TStrings);
14720: Procedure AddStrings1( Strings : TWideStrings);
14721: Procedure AddStringTerm( var sString : string; const sTerm : string; const sSeparator : string)
14722: Procedure AddToRecentDocs( const Filename : string)
14723: Procedure AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset)
14724: Procedure AllFunctionsList1Click( Sender : TObject)
14725: procedure AllObjectsList1Click(Sender: TObject);
14726: Procedure Allocate( AAllocateBytes : Integer)
14727: procedure AllResourceList1Click(Sender: TObject);
14728: Procedure AnsiAppend( var dst : AnsiString; const src : AnsiString)
14729: Procedure AnsiAssign( var dst : AnsiString; var src : AnsiString)
14730: Procedure AnsiDelete( var dst : AnsiString; index, count : Integer)
14731: Procedure AnsiFree( var s : AnsiString)
14732: Procedure AnsiFromWide( var dst : AnsiString; const src : WideString)
14733: Procedure AnsiInsert( var dst : AnsiString; const src : AnsiString; index : Integer)
14734: Procedure AnsiSetLength( var dst : AnsiString; len : Integer)
14735: Procedure AnsiString_to_stream( const Value : ansistring; Destin : TStream)
14736: Procedure AntiFreeze;
14737: Procedure APPEND
```

```
14738: Procedure Append( const S : WideString)
14739: procedure Append(S: string);
14740: Procedure AppendByte( var VBytes : TIdBytes; AByte : byte)
14741: Procedure AppendBytes( var VBytes : TIdBytes; AAdd : TIdBytes)
14742: Procedure AppendChunk( Val : OleVariant)
14743: Procedure AppendData( const Data : OleVariant; HitEOF : Boolean)
14744: Procedure AppendStr( var Dest : string; S : string)
14745: Procedure AppendString( var VBytes : TIdBytes; const AStr : String; ALength : Integer)
14746: Procedure ApplyRange
14747: procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
14748: Procedure Arrange( Code : TListArrangement)
14749: procedure Assert(expr : Boolean; const msg: string);
14750: procedure Assert2(expr : Boolean; const msg: string);
14751: Procedure Assign( AList : TCustomBucketList)
14752: Procedure Assign( Other : TObject)
14753: Procedure Assign( Source : TDragObject)
14754: Procedure Assign( Source : TPersistent)
14755: Procedure Assign(Source: TPersistent)
14756: procedure Assign2(mystring, mypath: string);
14757: Procedure AssignCurValues( Source : TDataSet);
14758: Procedure AssignCurValues1( const CurValues : Variant);
14759: Procedure ASSIGNFIELD( FIELD : TFIELD)
14760: Procedure ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT)
14761: Procedure AssignFile(var F: Text; FileName: string)
14762: procedure AssignFile(var F: TextFile; FileName: string)
14763: procedure AssignFileRead(var mystring, myfilename: string);
14764: procedure AssignFileWrite(mystring, myfilename: string);
14765: Procedure AssignTo( Other : TObject)
14766: Procedure AssignValues( Value : TParameters)
14767: Procedure ASSIGNVALUES( VALUE : TPARAMS)
14768: Procedure AssociateExtension( IconPath, ProgramName, Path, Extension : string)
14769: Procedure Base64_to_stream( const Base64 : ansistring; Destin : TStream)
14770: Procedure Base64ToVar( NatData : Pointer; const SoapData : WideString);
14771: Procedure Base64ToVar1( var V : Variant; const SoapData : WideString);
14772: Procedure BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
14773: Procedure BcdDivide( Dividend, Divisor : string; var bcdOut : TBcd);
14774: Procedure BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
14775: Procedure BcdDivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
14776: Procedure BcdDivide3( const Dividend : TBcd; const Divisor : string; var bcdOut : TBcd);
14777: Procedure BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
14778: Procedure BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
14779: Procedure BcdMultiply2( const bcdIn : TBcd; const StringIn : string; var bcdOut : TBcd);
14780: Procedure BcdMultiply3( StringIn1, StringIn2 : string; var bcdOut : TBcd);
14781: Procedure BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
14782: Procedure BcdToBytes( Value : TBcd; Bytes : array of byte)
14783: procedure Beep
14784: Procedure BeepOk
14785: Procedure BeepQuestion
14786: Procedure BeepHand
14787: Procedure BeepExclamation
14788: Procedure BeepAsterisk
14789: Procedure BeepInformation
14790: procedure BEGINDRAG(IMMEDIATE:BOOLEAN)
14791: Procedure BeginLayout
14792: Procedure BeginTimer( const Delay, Resolution : Cardinal)
14793: Procedure BeginUpdate
14794: procedure BeginUpdate;
14795: procedure BigScreen1Click(Sender: TObject);
14796: procedure BinToHex(Buffer: PChar; Text: PChar; BufSize: Integer);
14797: Procedure BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits : Boolean);
14798: Procedure BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits : Boolean);
14799: Procedure BitsToBooleans2( const Bits : Integer; var B : TBooleanArray; AllBits : Boolean);
14800: Procedure BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits : Boolean);
14801: Procedure BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord)
14802: Procedure BooleansToBits( var Dest : Byte; const B : TBooleanArray);
14803: Procedure BooleansToBits1( var Dest : Word; const B : TBooleanArray);
14804: Procedure BooleansToBits2( var Dest : Integer; const B : TBooleanArray);
14805: Procedure BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
14806: Procedure BreakPointMenuClick( Sender : TObject)
14807: procedure BRINGTOFRONT
14808: procedure BringToFront;
14809: Procedure btnBackClick( Sender : TObject)
14810: Procedure btnBrowseClick( Sender : TObject)
14811: Procedure BtnClick( Index : TNavigateBtn)
14812: Procedure btnLargeIconsClick( Sender : TObject)
14813: Procedure BuildAndSendRequest( AURI : TIdURI)
14814: Procedure BuildCache
14815: Procedure BurnMemory( var Buff, BuffLen : integer)
14816: Procedure BurnMemoryStream( Destructo : TMemoryStream)
14817: Procedure CalculateFirstSet
14818: Procedure Cancel
14819: procedure CancelDrag;
14820: Procedure CancelEdit
14821: procedure CANCELHINT
14822: Procedure CancelRange
14823: Procedure CancelUpdates
14824: Procedure CancelWriteBuffer
14825: Procedure Capture1(ADest:TStream;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool;
14826: Procedure Capture2( ADest : TStrings; const ADelim : string; const AIsRFCMessage : Boolean);
```

```
14827: Procedure Capture3(ADest:TStrings;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool
14828: procedure CaptureScreenFormat(vname: string; vextension: string);
14829: procedure CaptureScreenPNG(vname: string);
14830: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
14831: procedure CASCADE
14832: Procedure CastNativeToSoap(Info:PTypeInfo;var SoapData:WideString;NatData:Pointer;var IsNull: Boolean)
14833: Procedure CastSoapToVariant( SoapInfo : PTypeInfo; const SoapData : WideString; NatData : Pointer);
14834: Procedure cbPathClick( Sender : TObject)
14835: Procedure cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
14836: Procedure cedebugAfterExecute( Sender : TPSScript)
14837: Procedure cedebugBreakpoint( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
14838: Procedure cedebugCompile( Sender : TPSScript)
14839: Procedure cedebugExecute( Sender : TPSScript)
14840: Procedure cedebugIdle( Sender : TObject)
14841: Procedure cedebugLineInfo( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
14842: Procedure CenterHeight( const pc, pcParent : TControl)
14843: Procedure CenterDlg(AForm: TForm; MForm: TForm);   { Zentriert Forms }
14844: Procedure CenterForm(AForm: TForm; MForm: TForm);   { Zentriert Forms }
14845: Procedure Change
14846: procedure ChangeBiDiModeAlignment(var Alignment: TAlignment);
14847: Procedure Changed
14848: Procedure ChangeDir( const ADirName : string)
14849: Procedure ChangeDirUp
14850: Procedure ChangeEntryTransitions( oNewState : TniRegularExpressionState)
14851: Procedure ChangeLevelBy( Value : TChangeRange)
14852: Procedure ChDir(const s: string)
14853: Procedure Check(Status: Integer)
14854: Procedure CheckCommonControl( CC : Integer)
14855: Procedure CHECKFIELDNAME( const FIELDNAME : String)
14856: Procedure CHECKFIELDNAMES( const FIELDNAMES : String)
14857: Procedure CheckForDisconnect(const ARaiseExceptionIfDisconnected: boolean;const AIgnoreBuffer:bool)
14858: Procedure CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected : Boolean)
14859: Procedure CheckToken( T : Char)
14860: procedure CheckToken(t:char)
14861: Procedure CheckTokenSymbol( const S : string)
14862: procedure CheckTokenSymbol(s:string)
14863: Procedure CheckToolMenuDropdown( ToolButton : TToolButton)
14864: procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
14865: Procedure CIED65ToCIED50( var X, Y, Z : Extended)
14866: Procedure CIELABToBGR( const Source, Target : Pointer; const Count : Cardinal);
14867: procedure CipherFile1Click(Sender: TObject);
14868: Procedure Clear;
14869: Procedure Clear1Click( Sender : TObject)
14870: Procedure ClearColor( Color : TColor)
14871: Procedure CLEARITEM( AITEM : TMENUITEM)
14872: Procedure ClearMapping
14873: Procedure ClearSelection( KeepPrimary : Boolean)
14874: Procedure ClearWriteBuffer
14875: Procedure Click
14876: Procedure Close
14877: Procedure Close1Click( Sender : TObject)
14878: Procedure CloseDatabase( Database : TDatabase)
14879: Procedure CloseDataSets
14880: Procedure CloseDialog
14881: Procedure CloseFile(var F: Text);
14882: Procedure Closure
14883: Procedure CMYKToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
14884: Procedure CMYKToBGR1( const C, M, Y, K, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
14885: Procedure CodeCompletionList1Click( Sender : TObject)
14886: Procedure ColEnter
14887: Procedure Collapse
14888: Procedure Collapse( Recurse : Boolean)
14889: Procedure ColorRGBToHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word)
14890: Procedure CommaSeparatedToStringList( AList : TStrings; const Value : string)
14891: Procedure CommitFreeAndNil( var Transaction : TDBXTransaction)
14892: Procedure Compile1Click( Sender : TObject)
14893: procedure ComponentCount1Click(Sender: TObject);
14894: Procedure Compress(azipfolder, azipfile: string)
14895: Procedure DeCompress(azipfolder, azipfile: string)
14896: Procedure XZip(azipfolder, azipfile: string)
14897: Procedure XUnZip(azipfolder, azipfile: string)
14898: Procedure Connect( const ATimeout : Integer)
14899: Procedure Connect( Socket : TSocket)
14900: procedure Console1Click(Sender: TObject);
14901: Procedure Continue
14902: Procedure ContinueCount( var Counter : TJclCounter)
14903: procedure CONTROLDESTROYED(CONTROL:TCONTROL)
14904: Procedure ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : integer)
14905: Procedure ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : integer)
14906: Procedure ConvertImage(vsource, vdestination: string);
14907: // Ex. ConvertImage(Exepath+'my233_bmp.bmp',Exepath+'mypng111.png')
14908: Procedure ConvertToGray(Cnv: TCanvas);
14909: Procedure Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Integer; Length : Integer)
14910: Procedure Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Integer; Count : Integer);
14911: Procedure Copy1( Source : TBytes; Offset : Integer; Buffer : TValueBuffer; Count : Integer);
14912: Procedure CopyBytesToHostLongWord(const ASource:TIdBytes;const ASourceIndex:Integer;var VDest:LongWord)
14913: Procedure CopyBytesToHostWord( const ASource : TIdBytes; const ASourceIndex : Integer; var VDest : Word)
14914: Procedure CopyFrom( mbCopy : TMyBigInt)
14915: Procedure CopyMemoryStream( Source, Destination : TMemoryStream)
```

```
14916: procedure CopyRect(const Dest: TRect; Canvas: TCanvas;const Source: TRect);
14917: Procedure CopyTIdByteArray( const ASource : array of Byte; const ASourceIndex : Integer; var VDest : array
       of Byte; const ADestIndex : Integer; const ALength : Integer)
14918: Procedure CopyTIdBytes(const ASrc:TIdBytes;const ASrcIndex:Int;var VDest:TIdBytes;const ADestIdx:Int;const
       ALength:Int)
14919: Procedure CopyTIdCardinal( const ASource : Cardinal; var VDest : TIdBytes; const ADestIndex : Integer)
14920: Procedure CopyTIdInt64( const ASource : Int64; var VDest : TIdBytes; const ADestIndex : Integer)
14921: Procedure CopyTIdIPV6Address(const ASource:TIdIPv6Address; var VDest: TIdBytes; const ADestIndex : Integer)
14922: Procedure CopyTIdLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex : Integer)
14923: Procedure CopyTIdNetworkLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex:Integer)
14924: Procedure CopyTIdNetworkWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
14925: Procedure CopyTIdString(const ASource:String;var VDest:TIdBytes;const ADestIndex:Integer;ALength: Integer)
14926: Procedure CopyTIdWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
14927: Procedure CopyToClipboard
14928: Procedure CountParts
14929: Procedure CreateDataSet
14930: Procedure CreateEmptyFile( const FileName : string)
14931: Procedure CreateFileFromString( const FileName, Data : string)
14932: Procedure CreateFromDelta( Source : TPacketDataSet)
14933: procedure CREATEHANDLE
14934: Procedure CreateProcAsUser( const UserDomain, UserName, PassWord, CommandLine : string)
14935: Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const Environment:PChar)
14936: Procedure CreateTable
14937: Procedure CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString)
14938: procedure CSyntax1Click(Sender: TObject);
14939: Procedure CurrencyToComp( Value : Currency; var Result : Comp)
14940: Procedure CURSORPOSCHANGED
14941: procedure CutFirstDirectory(var S: String)
14942: Procedure DataBaseError(const Message: string)
14943: Procedure DateTimeToString( var Result : string; Format : string; DateTime : TDateTime);
14944: procedure DateTimeToString(var Result: string; const Format: string; DateTime: TDateTime)
14945: Procedure DateTimeToSystemTime( DateTime : TDateTime; var SystemTime : TSystemTime)
14946: procedure DateTimeToSystemTime(const DateTime: TDateTime; var SystemTime: TSystemTime);
14947: Procedure DBIError(errorCode: Integer)
14948: Procedure DebugOutput( const AText : string)
14949: Procedure DebugRun1Click( Sender : TObject)
14950: procedure Dec;
14951: Procedure DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word)
14952: procedure DecodeDate(const DateTime: TDateTime; var Year, Month, Day: Word);
14953: Procedure DecodeDateDay( const AValue : TDateTime; out AYear, ADayOfYear : Word)
14954: Procedure DecodeDateMonthWeek(const AValue:TDateTime;out AYear, AMonth,AWeekOfMonth,ADayOfWeek : Word)
14955: Procedure DecodeDateTime(const AValue:TDateTime;out AYear,AMonth,ADay,AHour,AMin,ASec,AMillSec:Word)
14956: Procedure DecodeDateWeek( const AValue : TDateTime; out AYear, AWeekOfYear, ADayOfWeek : Word)
14957: Procedure DecodeDayOfWeekInMonth(const AValue:TDateTime;out AYear,AMonth,ANthDayOfWeek,ADayOfWeek:Word)
14958: Procedure DecodeTime( DateTime : TDateTime; var Hour, Min, Sec, MSec : Word)
14959: procedure DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word);
14960: Procedure Decompile1Click( Sender : TObject)
14961: Procedure DefaultDrawColumnCell(const Rect : TRect; DataCol:Integer;Column:TColumn;State:TGridDrawState)
14962: Procedure DefaultDrawDataCell( const Rect : TRect; Field : TField; State : TGridDrawState)
14963: Procedure DeferLayout
14964: Procedure defFileread
14965: procedure DEFOCUSCONTROL(CONTROL:TWINCONTROL;REMOVING:BOOLEAN)
14966: Procedure DelayMicroseconds( const MicroSeconds : Integer)
14967: Procedure Delete
14968: Procedure Delete( const AFilename : string)
14969: Procedure Delete( const Index : Integer)
14970: Procedure DELETE( INDEX : INTEGER)
14971: Procedure Delete( Index : LongInt)
14972: Procedure Delete( Node : TTreeNode)
14973: procedure Delete(var s: AnyString; ifrom, icount: Longint);
14974: Procedure DeleteAlias( const Name : string)
14975: Procedure DeleteDriver( const Name : string)
14976: Procedure DeleteIndex( const Name : string)
14977: Procedure DeleteKey( const Section, Ident : String)
14978: Procedure DeleteRecords
14979: Procedure DeleteRecords( AffectRecords : TAffectRecords)
14980: Procedure DeleteString( var pStr : String; const pDelStr : string)
14981: Procedure DeleteTable
14982: procedure DelphiSite1Click(Sender: TObject);
14983: Procedure Deselect
14984: Procedure Deselect( Node : TTreeNode)
14985: procedure DestroyComponents
14986: Procedure DestroyHandle
14987: Procedure Diff( var X : array of Double)
14988: procedure Diff(var X: array of Double);
14989: procedure DISABLEALIGN
14990: Procedure DisableConstraints
14991: Procedure Disconnect
14992: Procedure Disconnect( Socket : TSocket)
14993: Procedure Dispose
14994: procedure Dispose(P: PChar)
14995: Procedure DivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
14996: Procedure DoKey( Key : TDBCtrlGridKey)
14997: Procedure DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
14998: Procedure DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
14999: Procedure Dormant
15000: Procedure DoubleToBcd1( const AValue : Double; var bcd : TBcd);
15001: Procedure DoubleToBytes( Value : Double; Bytes : array of byte)
15002: Procedure DoubleToComp( Value : Double; var Result : Comp)
```

```
15003: Procedure Draw( Canvas : TCanvas; X, Y, Index : Integer; Enabled : Boolean);
15004: procedure Draw(X, Y: Integer; Graphic: TGraphic);
15005: Procedure Draw1(Canvas:TCanvas; X,Y,
       Index:Int;ADrawingStyle:TDrawingStyle;AImageType:TImageType;Enabled:Bool);
15006: Procedure DrawArrow( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
15007: Procedure DrawCheck( ACanvas : TCanvas; Location : TPoint; Size : Integer; Shadow : Boolean)
15008: Procedure DrawChevron( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
15009: Procedure DrawColumnCell( const Rect : TRect; DataCol : Integer; Column : TColumn; State : TGridDrawState)
15010: procedure DrawFocusRect(const Rect: TRect);
15011: Procedure DrawHDIBToTBitmap( HDIB : THandle; Bitmap : TBitmap)
15012: Procedure DRAWMENUITEM( MENUITEM : TMENUITEM; ACANVAS : TCANVAS; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
15013: Procedure DrawOverlay(Canvas:TCanvas;X,Y:Integer;ImageIndex:Integer;Overlay:TOverlay;Enabled: Boolean);
15014: Procedure DrawOverlay1(Canvas:TCanvas; X,Y:Int; ImageIndex:Int; Overlay : TOverlay; ADrawingStyle :
       TDrawingStyle; AImageType : TImageType; Enabled : Boolean);
15015: procedure drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: integer);
15016: Procedure DrawPolyLine( const Canvas : TCanvas; var Points : TPointArray; const ClipRect : TRect)
15017: Procedure DropConnections
15018: Procedure DropDown
15019: Procedure DumpDescription( oStrings : TStrings)
15020: Procedure DumpStateTable( oStrings : TStrings)
15021: Procedure EDIT
15022: Procedure EditButtonClick
15023: Procedure EditFont1Click( Sender : TObject)
15024: procedure Ellipse(X1, Y1, X2, Y2: Integer);
15025: Procedure Ellipse1( const Rect : TRect);
15026: Procedure EMMS
15027: Procedure Encode( ADest : TStream)
15028: procedure ENDDRAG(DROP:BOOLEAN)
15029: Procedure EndEdit( Cancel : Boolean)
15030: Procedure EndTimer
15031: Procedure EndUpdate
15032: Procedure EraseSection( const Section : string)
15033: Procedure Error( const Ident : string)
15034: procedure Error(Ident:Integer)
15035: Procedure ErrorFmt( const Ident : string; const Args : array of const)
15036: Procedure ErrorStr( const Message : string)
15037: procedure ErrorStr(Message:String)
15038: Procedure Exchange( Index1, Index2 : Integer)
15039: procedure Exchange(Index1, Index2: Integer);
15040: Procedure Exec( FileName, Parameters, Directory : string)
15041: Procedure ExecProc
15042: Procedure ExecSQL( UpdateKind : TUpdateKind)
15043: Procedure Execute
15044: Procedure Execute( const CommandText : WideString; var Params, OwnerData : OleVariant)
15045: Procedure ExecuteAndWait( FileName : string; Visibility : Integer)
15046: Procedure ExecuteCommand(executeFile, paramstring: string)
15047: Procedure ExecuteShell(executeFile, paramstring: string)
15048: Procedure ExitThread(ExitCode: Integer); stdcall;
15049: Procedure ExitProcess(ExitCode: Integer); stdcall;
15050: Procedure Expand( AUserName : String; AResults : TStrings)
15051: Procedure Expand( Recurse : Boolean)
15052: Procedure ExportClipboard1Click( Sender : TObject)
15053: Procedure ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress)
15054: Procedure ExtractContentFields( Strings : TStrings)
15055: Procedure ExtractCookieFields( Strings : TStrings)
15056: Procedure ExtractFields( Separators, WhiteSpace : TSysCharSet; Content : PChar; Strings : TStrings)
15057: Procedure ExtractHeaderFields(Separ,
       WhiteSpace:TSysChSet;Content:PChar;Strings:TStrings;Decode:Bool;StripQuots:Bool)
15058: Procedure ExtractHTTPFields(Separators,WhiteSpace:
       TSysCharSet;Content:PChar;Strings:TStrings;StripQuotes:Bool)
15059: Procedure ExtractQueryFields( Strings : TStrings)
15060: Procedure FastDegToGrad
15061: Procedure FastDegToRad
15062: Procedure FastGradToDeg
15063: Procedure FastGradToRad
15064: Procedure FastRadToDeg
15065: Procedure FastRadToGrad
15066: Procedure FileClose( Handle : Integer)
15067: Procedure FileClose(handle: integer)
15068: procedure FilesFromWildcard(Dir,Mask:string;var Files:TStringList;Subdirs,ShowDirs,Multitasking:Bool)
15069: Procedure FileStructure( AStructure : TIdFTPDataStructure)
15070: Procedure FillByte2(var X: Byte ; count: integer; value: byte)
15071: Procedure FillBytes( var VBytes : TIdBytes; const ACount : Integer; const AValue : Byte)
15072: Procedure FillChar( Buffer : TRecordBuffer; Length : Integer; value : Byte)
15073: Procedure FillChar2(var X: PChar ; count: integer; value: char)
15074: Procedure FillCharS(var p: string; count: integer; value: char); //fix3.8
15075: Procedure FillIPList
15076: procedure FillRect(const Rect: TRect);
15077: Procedure FillTStrings( AStrings : TStrings)
15078: Procedure FilterOnBookmarks( Bookmarks : array of const)
15079: procedure FinalizePackage(Module: HMODULE)
15080: procedure FindClose;
15081: procedure FindClose2(var F: TSearchRec)
15082: Procedure FindMatches( const sString : string; xNotify : TniRegularExpressionMatchFoundEvent);
15083: Procedure FindMatches1(const sString:string;iStart:integer;xNotify:TniRegularExpressionMatchFoundEvent);
15084: Procedure FindNearest( const KeyValues : array of const)
15085: Procedure FinishContext
15086: Procedure FIRST
15087: Procedure FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float)
```

```
15088: Procedure FloatToDecimal(var Result:TFloatRec;const Value:extend;ValueType:TFloatValue;Precis,Decs:Int);
15089: Procedure FloodFill( X, Y : Integer; Color : TColor; FillStyle : TFillStyle)
15090: Procedure FlushSchemaCache( const TableName : string)
15091: Procedure FmtStr(var Result: string; const Format: string; const Args: array of const)
15092: procedure FOCUSCONTROL(CONTROL:TWINCONTROL)
15093: Procedure Form1Close( Sender : TObject; var Action : TCloseAction)
15094: Procedure FormActivate( Sender : TObject)
15095: procedure FormatLn(const format: String; const args: array of const);  //alias
15096: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
15097: Procedure FormCreate( Sender : TObject)
15098: Procedure FormDestroy( Sender : TObject)
15099: Procedure FormKeyPress( Sender : TObject; var Key : Char)
15100: procedure FormOutput1Click(Sender: TObject);
15101: Procedure FormToHtml( Form : TForm; Path : string)
15102: procedure FrameRect(const Rect: TRect);
15103: Procedure Frame3D(Canvas:TCanvas; var Rect:TRect; TopColor,BottomColor:TColor; Width : Integer)
15104: Procedure NotebookHandlesNeeded( Notebook : TNotebook)
15105: Procedure Free( Buffer : TRecordBuffer)
15106: Procedure Free( Buffer : TValueBuffer)
15107: Procedure Free;
15108: Procedure FreeAndNil(var Obj:TObject)
15109: Procedure FreeImage
15110: procedure FreeMem(P: PChar; Size: Integer)
15111: Procedure FreeTreeData( Tree : TUpdateTree)
15112: Procedure Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Integer)
15113: Procedure FullCollapse
15114: Procedure FullExpand
15115: Procedure GenerateDPB(sl: TStrings; var DPB: string; var DPBLength: Short);  //InterBase
15116: Procedure GenerateTPB(sl: TStrings; var TPB: string; var TPBLength: Short);
15117: Procedure OUTPUTXML( SQLOBJECT : TIBSQL; OUTPUTOBJECT : TIBOUTPUTXML)
15118: Procedure Get1( AURL : string; const AResponseContent : TStream);
15119: Procedure Get1( const ASourceFile : string; ADest : TStream; AResume : Boolean);
15120: Procedure Get2(const ASourceFile,ADestFile: string;const ACanOverwrite: boolean; AResume: Boolean);
15121: Procedure GetAliasNames( List : TStrings)
15122: Procedure GetAliasParams( const AliasName : string; List : TStrings)
15123: Procedure GetApplicationsRunning( Strings : TStrings)
15124: Procedure GetCommandTypes( List : TWideStrings)
15125: Procedure GetConfigParams( const Path, Section : string; List : TStrings)
15126: Procedure GetConnectionNames( List : TStrings; Driver : string; DesignMode : Boolean)
15127: Procedure GetConvFamilies( out AFamilies : TConvFamilyArray)
15128: Procedure GetConvTypes( const AFamily : TConvFamily; out ATypes : TConvTypeArray)
15129: Procedure GetDatabaseNames( List : TStrings)
15130: Procedure GetDataPacket( DataSet : TDataSet; var RecsOut : Integer; out Data : OleVariant)
15131: Procedure GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize: longWORD; var ImageSize : longWORD)
15132: Procedure GetDir(d: byte; var s: string)
15133: Procedure GetDirList( const Search : string; List : TStrings; Recursive : Boolean)
15134: Procedure GetDriverNames( List : TStrings)
15135: Procedure GetDriverNames( List : TStrings; DesignMode : Boolean)
15136: Procedure GetDriverParams( const DriverName : string; List : TStrings)
15137: Procedure GetEMails1Click( Sender : TObject)
15138: Procedure getEnvironmentInfo;
15139: Function getEnvironmentString: string;
15140: Procedure GetFieldNames( const DatabaseName, TableName : string; List : TStrings)
15141: Procedure GetFieldNames( const TableName : string; List : TStrings)
15142: Procedure GetFieldNames( const TableName : string; List : TStrings);
15143: Procedure GetFieldNames( const TableName : WideString; List : TWideStrings);
15144: Procedure GETFIELDNAMES( LIST : TSTRINGS)
15145: Procedure GetFieldNames1( const TableName : string; List : TStrings);
15146: Procedure GetFieldNames1( const TableName : string; SchemaName : string; List : TStrings);
15147: Procedure GetFieldNames2( const TableName : WideString;SchemaName : WideString; List : TWideStrings);
15148: Procedure GetFieldNames3( const TableName : WideString; List : TWideStrings);
15149: Procedure GetFileAttributeList( const Items : TStrings; const Attr : Integer)
15150: Procedure GetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
15151: Procedure GetFMTBcd( Buffer : TRecordBuffer; var value : TBcd)
15152: Procedure GetFormatSettings
15153: Procedure GetFromDIB( var DIB : TBitmapInfo)
15154: Procedure GetFromHDIB( HDIB : HBitmap)
15155: Procedure GetIcon( Index : Integer; Image : TIcon);
15156: Procedure GetIcon1(Index : Integer; Image:TIcon; ADrawingStyle:TDrawingStyle; AImageType:TImageType);
15157: Procedure GetIndexInfo( IndexName : string)
15158: Procedure GetIndexNames( const TableName, SchemaName : string; List : TStrings);
15159: Procedure GetIndexNames( List : TStrings)
15160: Procedure GetIndexNames1( const TableName : WideString; List : TWideStrings);
15161: Procedure GetIndexNames2( const TableName, SchemaName : WideString; List : TWideStrings);
15162: Procedure GetIndexNames4( const TableName : string; List : TStrings);
15163: Procedure GetInternalResponse
15164: Procedure GETITEMNAMES( LIST : TSTRINGS)
15165: procedure GetMem(P: PChar; Size: Integer)
15166: Procedure GETOLE2ACCELERATORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:INTEGER;GROUPS:array of INTEGER)
15167: procedure GetPackageDescription(ModuleName: PChar): string)
15168: Procedure GetPackageNames( List : TStrings);
15169: Procedure GetPackageNames1( List : TWideStrings);
15170: Procedure GetParamList( List : TList; const ParamNames : WideString)
15171: Procedure GetProcedureNames( List : TStrings);
15172: Procedure GetProcedureNames( List : TWideStrings);
15173: Procedure GetProcedureNames1( const PackageName : string; List : TStrings);
15174: Procedure GetProcedureNames1( List : TStrings);
15175: Procedure GetProcedureNames2( const PackageName, SchemaName : string; List : TStrings);
15176: Procedure GetProcedureNames3( List : TWideStrings);
```

```
15177: Procedure GetProcedureNames4( const PackageName : Widestring; List : TWideStrings);
15178: Procedure GetProcedureNames5( const PackageName, SchemaName : WideString; List : TWideStrings);
15179: Procedure GetProcedureParams( ProcedureName : WideString; List : TList);
15180: Procedure GetProcedureParams1( ProcedureName, PackageName : WideString; List : TList);
15181: Procedure GetProcedureParams2( ProcedureName, PackageName, SchemaName : Widestring; List : TList);
15182: Procedure GetProviderNames( Names : TWideStrings);
15183: Procedure GetProviderNames( Proc : TGetStrProc)
15184: Procedure GetProviderNames1( Names : TStrings);
15185: procedure GetQrCode2(Width,Height:Word; Correct_Level: string; const Data:string; apath: string);
15186: procedure GetQrCode3(Width,Height:Word; Correct_Level: string; const Data:string; apath: string);//no auto
       open image
15187: Function GetQrCode4(Width,Height:Word;Correct_Level:string; const
       Data:string;aformat:string):TLinearBitmap;
15188: Procedure GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte)
15189: Procedure GetSchemaNames( List : TStrings);
15190: Procedure GetSchemaNames1( List : TWideStrings);
15191: Procedure GetSessionNames( List : TStrings)
15192: Procedure GetStoredProcNames( const DatabaseName : string; List : TStrings)
15193: Procedure GetStrings( List : TStrings)
15194: Procedure GetSystemTime; stdcall;
15195: Procedure GetTableNames(const DatabaseName,Pattern:string;Extensions,SystemTables:Boolean;List:TStrings)
15196: Procedure GetTableNames( List : TStrings; SystemTables : Boolean)
15197: Procedure GetTableNames( List : TStrings; SystemTables : Boolean);
15198: Procedure GetTableNames( List : TWideStrings; SystemTables : Boolean);
15199: Procedure GetTableNames1( List : TStrings; SchemaName : WideString; SystemTables : Boolean);
15200: Procedure GetTableNames1( List : TStrings; SystemTables : Boolean);
15201: Procedure GetTableNames2( List : TWideStrings; SchemaName : WideString; SystemTables : Boolean);
15202: Procedure GetTransitionsOn( cChar : char; oStateList : TList)
15203: Procedure GetVisibleWindows( List : Tstrings)
15204: Procedure GoBegin
15205: Procedure GotoCurrent( DataSet : TCustomClientDataSet)
15206: Procedure GotoCurrent( Table : TTable)
15207: procedure GotoEnd1Click(Sender: TObject);
15208: Procedure GotoNearest
15209: Procedure GradientFillCanvas(const ACanvas:TCanvas;const AStartCol,AEndCol:TColor;const ARect:TRect;const
       Direction: TGradientDirection)
15210: Procedure HandleException( E : Exception; var Handled : Boolean)
15211: procedure HANDLEMESSAGE
15212: procedure HandleNeeded;
15213: Procedure Head( AURL : string)
15214: Procedure Help( var AHelpContents : TStringList; ACommand : String)
15215: Procedure HexToBinary( Stream : TStream)
15216: procedure HexToBinary(Stream:TStream)
15217: Procedure HideDragImage
15218: Procedure HideFormCaption( FormHandle : THandle; Hide : Boolean)
15219: Procedure HideTraybar
15220: Procedure HideWindowForSeconds(secs: integer);   {//3 seconds}
15221: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm);   {//3 seconds}
15222: Procedure HookOSExceptions
15223: Procedure HookSignal( RtlSigNum : Integer)
15224: Procedure HSLToRGB( const H, S, L : Single; out R, G, B : Single);
15225: Procedure HTMLSyntax1Click( Sender : TObject)
15226: Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
15227: Procedure IFPS3ClassesPlugin1ExecImport( Sender : TObject; Exec : TPSExec; x : TPSRuntimeClassImporter)
15228: Procedure ImportfromClipboard1Click( Sender : TObject)
15229: Procedure ImportfromClipboard2Click( Sender : TObject)
15230: Procedure IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Integer)
15231: procedure Incb(var x: byte);
15232: Procedure Include1Click( Sender : TObject)
15233: Procedure IncludeOFF;  //preprocessing
15234: Procedure IncludeON;
15235: procedure Info1Click(Sender: TObject);
15236: Procedure InitAltRecBuffers( CheckModified : Boolean)
15237: Procedure InitContext( Request : TWebRequest; Response : TWebResponse)
15238: Procedure InitContext(WebModuleList:TAbstractWebModuleList;Request:TWebRequest;Response:TWebResponse)
15239: Procedure InitData( ASource : TDataSet)
15240: Procedure InitDelta( ADelta : TPacketDataSet);
15241: Procedure InitDelta1( const ADelta : OleVariant);
15242: Procedure InitErrorPacket( E : EUpdateError; Response : TResolverResponse)
15243: Procedure Initialize
15244: procedure InitializePackage(Module: HMODULE)
15245: Procedure INITIATEACTION
15246: Procedure initHexArray(var hexn: THexArray);  //THexArray', 'array[0..15] of char;'
15247: Procedure InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet)
15248: Procedure InitModule( AModule : TComponent)
15249: Procedure InitStdConvs
15250: Procedure InitTreeData( Tree : TUpdateTree)
15251: Procedure INSERT
15252: Procedure Insert( Index : Integer; AClass : TClass)
15253: Procedure Insert( Index : Integer; AComponent : TComponent)
15254: Procedure Insert( Index : Integer; AObject : TObject)
15255: Procedure Insert( Index : Integer; const S : WideString)
15256: Procedure Insert( Index : Integer; Image, Mask : TBitmap)
15257: Procedure Insert(Index: Integer; const S: string);
15258: procedure Insert(Index: Integer; S: string);
15259: procedure Insert(s: AnyString; var s2: AnyString; iPos: Longint);
15260: procedure InsertComponent(AComponent:TComponent)
15261: procedure InsertControl(AControl: TControl);
15262: Procedure InsertIcon( Index : Integer; Image : TIcon)
```

```
15263: Procedure InsertMasked( Index : Integer; Image : TBitmap; MaskColor : TColor)
15264: Procedure InsertObject( Index : Integer; const S : WideString; AObject : TObject)
15265: procedure InsertObject(Index:Integer;S:String;AObject:TObject)
15266: Procedure Int16ToBytes( Value : SmallInt; Bytes : array of byte)
15267: Procedure Int32ToBytes( Value : Integer; Bytes : array of byte)
15268: Procedure Int64ToBytes( Value : Int64; Bytes : array of byte)
15269: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
15270: Procedure InternalBeforeResolve( Tree : TUpdateTree)
15271: Procedure InvalidateModuleCache
15272: Procedure InvalidateTitles
15273: Procedure InvalidDateDayError( const AYear, ADayOfYear : Word)
15274: Procedure InvalidDateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
15275: Procedure InvalidDateTimeError(const AYear,AMth,ADay,AHour,AMin,ASec,AMilSec:Word;const
       ABaseDate:TDateTime)
15276: Procedure InvalidDateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
15277: Procedure InvalidDayOfWeekInMonthError( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word)
15278: procedure JavaSyntax1Click(Sender: TObject);
15279: Procedure JclLocalesInfoList( const Strings : TStrings; InfoType : Integer)
15280: Procedure KillDataChannel
15281: Procedure Largefont1Click( Sender : TObject)
15282: Procedure LAST
15283: Procedure LaunchCpl( FileName : string)
15284: Procedure Launch( const AFile : string)
15285: Procedure LaunchFile( const AFile : string)
15286: Procedure LetFileList(FileList: TStringlist; apath: string);
15287: Procedure lineToNumber( xmemo : String; met : boolean)
15288: Procedure ListViewCustomDrawItem(Sender:TCustomListView;Item:TListItem;State:TCustomDrawState;var
       DefaultDraw:Bool)
15289: Procedure ListViewCustomDrawSubItem( Sender : TCustomListView; Item : TListItem; SubItem : Integer; State
       : TCustomDrawState; var DefaultDraw : Boolean)
15290: Procedure ListViewData( Sender : TObject; Item : TListItem)
15291: Procedure ListViewDataFind(Sender:TObject; Find : TItemFind; const FindString : String; const FindPosition
       : TPoint; FindData:Pointer; StartIndex:Integer;Direction:TSearchDirection;Wrap:Boolean;var Index: Integer)
15292: Procedure ListViewDataHint( Sender : TObject; StartIndex, EndIndex : Integer)
15293: Procedure ListViewDblClick( Sender : TObject)
15294: Procedure ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
15295: Procedure ListDLLExports(const FileName: string; List: TStrings);
15296: Procedure Load( const WSDLFileName : WideString; Stream : TMemoryStream)
15297: procedure LoadBytecode1Click(Sender: TObject);
15298: procedure LoadFilefromResource(const FileName: string; ms: TMemoryStream);
15299: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HP)
15300: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
15301: Procedure LoadFromFile( AFileName : string)
15302: Procedure LoadFromFile( const AFileName : string; const AHeadersOnly : Boolean)
15303: Procedure LoadFromFile( const FileName : string)
15304: Procedure LOADFROMFILE( const FILENAME : String; BLOBTYPE : TBLOBTYPE)
15305: Procedure LoadFromFile( const FileName : string; DataType : TDataType)
15306: Procedure LoadFromFile( const FileName : WideString)
15307: Procedure LoadFromFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
15308: Procedure LoadFromFile(const AFileName: string)
15309: procedure LoadFromFile(FileName: string);
15310: procedure LoadFromFile(FileName:String)
15311: Procedure LoadFromResourceID( Instance : THandle; ResID : Integer)
15312: Procedure LoadFromResourceName( Instance : THandle; const ResName : String)
15313: Procedure LoadFromStream( AStream : TStream; const AHeadersOnly : Boolean)
15314: Procedure LoadFromStream( const Stream : TStream)
15315: Procedure LoadFromStream( S : TStream)
15316: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
15317: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
15318: Procedure LoadFromStream( Stream : TStream)
15319: Procedure LOADFROMSTREAM( STREAM : TSTREAM; BLOBTYPE : TBLOBTYPE)
15320: Procedure LoadFromStream( Stream : TStream; DataType : TDataType)
15321: procedure LoadFromStream(Stream: TStream);
15322: Procedure LoadFromStream1( Stream : TSeekableStream; const FormatExt : string);
15323: Procedure LoadFromStream2( Stream : TStream; const FormatExt : string);
15324: Procedure LoadFromStrings( AStrings : TStrings; const MimeSeparator : Char)
15325: Procedure LoadLastFile1Click( Sender : TObject)
15326: { LoadIcoToImage loads two icons from resource named NameRes,
15327:   into two image lists ALarge and ASmall}
15328: Procedure LoadIcoToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes: string);
15329: Procedure LoadMemo
15330: Procedure LoadParamsFromIniFile( FFileName : WideString)
15331: Procedure Lock
15332: Procedure Login
15333: Procedure MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
15334: Procedure MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
15335: Procedure MakeCaseInsensitive
15336: Procedure MakeDeterministic( var bChanged : boolean)
15337: Procedure MakeGrayPal( var Palette, ColorCount : Integer)
15338: // type TVolumeLevel = 0..127;  , savaFilePath as C:\MyFile.wav
15339: Procedure MakeSound(Frequency{Hz}, Duration{mSec}: Int; Volume: TVolumeLevel; savefilePath: string);
15340: Procedure MakeComplexSound(N:integer;freqlist:TStrings;Duration{mSec}:Int;pinknoise:bool;Volume:Byte);
15341: Procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
15342: Procedure SetRectComplexFormatStr( const S : string)
15343: Procedure SetPolarComplexFormatStr( const S : string)
15344: Procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
15345: Procedure MakeVisible
15346: Procedure MakeVisible( PartialOK : Boolean)
15347: Procedure Manual1Click( Sender : TObject)
```

```
15348: Procedure MarkReachable
15349: Procedure maXbox;  //shows the exe version data in a win box
15350: Procedure MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
15351: Procedure Memo1Change( Sender : TObject)
15352: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Int;var
       Action:TSynReplaceAction)
15353: Procedure Memo1SpecialLineColors(Sender:TObject;Line:Integer; var Special:Boolean;var FG,BG: TColor)
15354: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
15355: procedure Memory1Click(Sender: TObject);
15356: Procedure MERGE( MENU : TMAINMENU)
15357: Procedure MergeChangeLog
15358: procedure MINIMIZE
15359: Procedure MinimizeMaxbox;
15360: Procedure MkDir(const s: string)
15361: Procedure mnuPrintFont1Click( Sender : TObject)
15362: procedure ModalStarted
15363: Procedure Modified
15364: Procedure ModifyAlias( Name : string; List : TStrings)
15365: Procedure ModifyDriver( Name : string; List : TStrings)
15366: Procedure MomentSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4, Skew,Kurtosis:Extended)
15367: Procedure MouseToCell( X, Y : Integer; var ACol, ARow : Longint)
15368: Procedure Move( CurIndex, NewIndex : Integer)
15369: procedure Move(CurIndex, NewIndex: Integer);
15370: procedure Move2(const Source: TByteArray; var Dest: TByteArray; Count: Integer)
15371: Procedure MoveChars(const ASource: String; ASourceStart:int;var ADest: String; ADestStart,ALen:integer)
15372: Procedure moveCube( o : TMyLabel)
15373: Procedure MoveTo( Destination : LongInt; AttachMode : TAttachMode)
15374: procedure MoveTo(X, Y: Integer);
15375: procedure MoveWindowOrg(DC: HDC; DX, DY: Integer);
15376: Procedure MovePoint(var x,y:Extended; const angle:Extended);
15377: Procedure Multiply( Multiplier1, Multiplier2 : TMyBigInt);
15378: Procedure Multiply1( Multiplier1 : TMyBigInt; Multiplier2 : Integer);
15379: Procedure MsgAbout(Handle:Int;const Msg,Caption:string;const IcoName:string ='MAINICON';Flags:DWORD=MB_OK);
15380: Procedure mxButton(x,y,width,height,top,left,ahandle: integer);
15381: Procedure New( Width, Height : Integer; PixFormat : TPixelFormat)
15382: procedure New(P: PChar)
15383: procedure New1Click(Sender: TObject);
15384: procedure NewInstance1Click(Sender: TObject);
15385: Procedure NEXT
15386: Procedure NextMonth
15387: Procedure Noop
15388: Procedure NormalizePath( var APath : string)
15389: procedure ObjectBinaryToText(Input, Output: TStream)
15390: procedure ObjectBinaryToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
15391: procedure ObjectResourceToText(Input, Output: TStream)
15392: procedure ObjectResourceToText1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
15393: procedure ObjectTextToBinary(Input, Output: TStream)
15394: procedure ObjectTextToBinary1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
15395: procedure ObjectTextToResource(Input, Output: TStream)
15396: procedure ObjectTextToResource1(Input, Output: TStream;var OriginalFormat: TStreamOriginalFormat)
15397: Procedure Open( const Name, Address, Service : string; Port : Word; Block : Boolean)
15398: Procedure Open( const UserID : WideString; const Password : WideString);
15399: Procedure Open;
15400: Procedure open1Click( Sender : TObject)
15401: Procedure OpenCdDrive
15402: Procedure OpenCloseCdDrive( OpenMode : Boolean; Drive : Char)
15403: Procedure OpenCurrent
15404: Procedure OpenFile(vfilenamepath: string)
15405: Procedure OpenDirectory1Click( Sender : TObject)
15406: Procedure OpenIndexFile( const IndexName : string)
15407: Procedure OpenSchema(const Schema:TSchemaInf;const Restricts:OleVar;const
       SchemaID:OleVariant;DataSet:TADODataSet)
15408: Procedure OpenWriteBuffer( const AThreshhold : Integer)
15409: Procedure OptimizeMem
15410: Procedure Options1( AURL : string);
15411: Procedure OutputDebugString(lpOutputString : PChar)
15412: Procedure PackBuffer
15413: Procedure Paint
15414: Procedure PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch : Boolean)
15415: Procedure PaintToTBitmap( Target : TBitmap)
15416: Procedure PaletteChanged
15417: Procedure ParentBiDiModeChanged
15418: Procedure PARENTBIDIMODECHANGED( ACONTROL : TOBJECT)
15419: Procedure PasteFromClipboard;
15420: Procedure PasteImage( Source : TLinearBitmap; X, Y : Integer)
15421: Procedure PathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
15422: Procedure PerformEraseBackground(Control: TControl; DC: HDC);
15423: Procedure PError( Text : string)
15424: procedure Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
15425: procedure Pie(X1:Integer;Y1:Integer;X2:Integer;Y2:Integer;X3:Int;Y3:Integer;X4:Integer;Y4:Integer);
15426: Procedure Play( FromFrame, ToFrame : Word; Count : Integer)
15427: procedure playmp3(mpath: string);
15428: Procedure PlayMP31Click( Sender : TObject)
15429: Procedure PointCopy( var Dest : TPoint; const Source : TPoint)
15430: Procedure PointMove( var P : TPoint; const DeltaX, DeltaY : Integer)
15431: procedure PolyBezier(const Points: array of TPoint);
15432: procedure PolyBezierTo(const Points: array of TPoint);
15433: procedure Polygon(const Points: array of TPoint);
15434: procedure Polyline(const Points: array of TPoint);
```

```
15435: Procedure Pop
15436: Procedure POPULATEOLE2MENU(SHAREDMENU: HMENU;GROUPS: array of INT; var WIDTHS : array of LONGINT)
15437: Procedure PopulationVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
15438: Procedure POPUP( X, Y : INTEGER)
15439: Procedure PopupURL(URL : WideString);
15440: Procedure POST
15441: Procedure Post4( AURL : string; const ASource : TStrings; const AResponseContent : TStream);
15442: Procedure Post5( AURL : string; const ASource, AResponseContent : TStream);
15443: Procedure Post6( AURL : string; const ASource : TIdMultiPartFormDataStream; AResponseContent : TStream);
15444: Procedure PostUser( const Email, FirstName, LastName : WideString)
15445: Procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
15446: procedure Pred(X: int64);
15447: Procedure Prepare
15448: Procedure PrepareStatement
15449: Procedure PreProcessXML( AList : TStrings)
15450: Procedure PreventDestruction
15451: Procedure Print( const Caption : string)
15452: procedure PrintBitmap(aGraphic: TGraphic; Title: string);
15453: procedure printf(const format: String; const args: array of const);
15454: Procedure PrintList(Value: TStringList);
15455: Procedure PrintImage(aValue:TBitmap;Style:TBitmapStyle);//TBitmapStyle=(bsNormal,bsCentered,bsStretched)
15456: Procedure Printout1Click( Sender : TObject)
15457: Procedure ProcessHeaders
15458: Procedure PROCESSMENUCHAR( var MESSAGE : TWMMENUCHAR)
15459: Procedure ProcessMessage( AMsg : TIdMessage; AHeaderOnly : Boolean);
15460: Procedure ProcessMessage1( AMsg : TIdMessage; const AStream : TStream; AHeaderOnly : Boolean);
15461: Procedure ProcessMessage2( AMsg : TIdMessage; const AFilename : string; AHeaderOnly : Boolean);
15462: Procedure ProcessMessagesOFF;  //application.processmessages
15463: Procedure ProcessMessagesON;
15464: Procedure ProcessPath(const EditText:string;var Drive:Char; var DirPart:string; var FilePart : string)
15465: Procedure ProcessPath1(const EditText:string;var Drive:Char; var DirPart:string; var FilePart:string);
15466: Procedure Proclist Size is: 3797 /1415
15467: Procedure procMessClick( Sender : TObject)
15468: Procedure PSScriptCompile( Sender : TPSScript)
15469: Procedure PSScriptExecute( Sender : TPSScript)
15470: Procedure PSScriptLine( Sender : TObject)
15471: Procedure Push( ABoundary : string)
15472: procedure PushItem(AItem: Pointer)
15473: Procedure Put2( AURL : string; const ASource, AResponseContent : TStream);
15474: Procedure Put2( const ASourceFile : string; const ADestFile : string; const AAppend : boolean);
15475: procedure PutLinuxLines(const Value: string)
15476: Procedure Quit
15477: Procedure RaiseConversionError( const AText : string);
15478: Procedure RaiseConversionError1( const AText : string; const AArgs : array of const);
15479: Procedure RaiseConversionRegError( AFamily : TConvFamily; const ADescription : string)
15480: procedure RaiseException(Ex: TIFException; Param: String);
15481: Procedure RaiseExceptionForLastCmdResult;
15482: procedure RaiseLastException;
15483: procedure RaiseException2;
15484: Procedure RaiseLastOSError
15485: Procedure RaiseLastWin32;
15486: procedure RaiseLastWin32Error)
15487: Procedure RaiseListError( const ATemplate : string; const AData : array of const)
15488: Procedure RandomFillStream( Stream : TMemoryStream)
15489: procedure randomize;
15490: Procedure Rasterize( Rasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect)
15491: Procedure RCS
15492: Procedure Read( Socket : TSocket)
15493: Procedure ReadBlobData
15494: procedure ReadBuffer(Buffer:String;Count:LongInt)
15495: procedure ReadOnly1Click(Sender: TObject);
15496: Procedure ReadSection( const Section : string; Strings : TStrings)
15497: Procedure ReadSections( Strings : TStrings)
15498: Procedure ReadSections( Strings : TStrings);
15499: Procedure ReadSections1( const Section : string; Strings : TStrings);
15500: Procedure ReadSectionValues( const Section : string; Strings : TStrings)
15501: Procedure ReadStream( AStream : TStream; AByteCount:LongInt; const AReadUntilDisconnect : boolean)
15502: Procedure ReadStrings( ADest : TStrings; AReadLinesCount : Integer)
15503: Procedure ReadVersion2(aFileName: STRING; aVersion : TStrings);
15504: Function ReadVersion(aFileName: STRING; aVersion : TStrings): boolean;
15505: Procedure Realign;
15506: procedure Rectangle(X1, Y1, X2, Y2: Integer);
15507: Procedure Rectangle1( const Rect : TRect);
15508: Procedure RectCopy( var Dest : TRect; const Source : TRect)
15509: Procedure RectFitToScreen( var R : TRect)
15510: Procedure RectGrow( var R : TRect; const Delta : Integer)
15511: Procedure RectGrowX( var R : TRect; const Delta : Integer)
15512: Procedure RectGrowY( var R : TRect; const Delta : Integer)
15513: Procedure RectMove( var R : TRect; const DeltaX, DeltaY : Integer)
15514: Procedure RectMoveTo( var R : TRect; const X, Y : Integer)
15515: Procedure RectNormalize( var R : TRect)
15516: //  TFileCallbackProcedure = procedure(filename:string);
15517: Procedure RecurseDirectory(Dir: String;IncludeSubs: boolean;callback: TFileCallbackProcedure);
15518: Procedure RecurseDirectory2(Dir: String; IncludeSubs : boolean);
15519: Procedure RedirectTransition(oOldState:TniRegularExpressionState; oNewState : TniRegularExpressionState)
15520: Procedure Refresh;
15521: Procedure RefreshData( Options : TFetchOptions)
15522: Procedure REFRESHLOOKUPLIST
15523: Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
```

```
15524: Procedure RegisterChanges( Value : TChangeLink)
15525: Procedure RegisterConversionFormat( const AExtension : string; AConversionClass : TConversionClass)
15526: Procedure RegisterFileFormat( const AExtension, ADescription : string; AGraphicClass : TGraphicClass)
15527: Procedure RegisterFileFormat(Extension,AppID:string;Description:string;Executable:string;IconIndex:Int)
15528: Procedure ReInitialize( ADelay : Cardinal)
15529: procedure RELEASE
15530: Procedure Remove( const AByteCount : integer)
15531: Procedure REMOVE( FIELD : TFIELD)
15532: Procedure REMOVE( ITEM : TMENUITEM)
15533: Procedure REMOVE( POPUP : TPOPUPMENU)
15534: Procedure RemoveAllPasswords
15535: procedure RemoveComponent(AComponent:TComponent)
15536: Procedure RemoveDir( const ADirName : string)
15537: Procedure RemoveLambdaTransitions( var bChanged : boolean)
15538: Procedure REMOVEPARAM( VALUE : TPARAM)
15539: Procedure RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharset);
15540: Procedure RemoveTransitionTo1( oState : TniRegularExpressionState);
15541: Procedure Rename( const ASourceFile, ADestFile : string)
15542: Procedure Rename( const FileName : string; Reload : Boolean)
15543: Procedure RenameTable( const NewTableName : string)
15544: Procedure Replace( Index : Integer; Image, Mask : TBitmap)
15545: Procedure Replace1Click( Sender : TObject)
15546: Procedure ReplaceDate( var DateTime : TDateTime; NewDate : TDateTime)
15547: procedure ReplaceDate(var DateTime: TDateTime; const NewDate: TDateTime))
15548: Procedure ReplaceIcon( Index : Integer; Image : TIcon)
15549: Procedure ReplaceMasked( Index : Integer; NewImage : TBitmap; MaskColor : TColor)
15550: Procedure ReplaceTime( var DateTime : TDateTime; NewTime : TDateTime)
15551: procedure ReplaceTime(var DateTime: TDateTime; const NewTime: TDateTime);
15552: Procedure Requery( Options : TExecuteOptions)
15553: Procedure Reset
15554: Procedure Reset1Click( Sender : TObject)
15555: Procedure ResizeCanvas( XSiz, YSiz, XPos, YPos : Integer; Color : TColor)
15556: procedure ResourceExplore1Click(Sender: TObject);
15557: Procedure RestoreContents
15558: Procedure RestoreDefaults
15559: Procedure RestoreOtherInstance( MainFormClassName, MainFormCaption : string)
15560: Procedure RetrieveHeaders
15561: Procedure RevertRecord
15562: Procedure RGBAToBGRA( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal)
15563: Procedure RGBToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
15564: Procedure RGBToBGR1( const R, G, B, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
15565: Procedure RGBToHSL( const R, G, B : Single; out H, S, L : Single);
15566: Procedure RGBToHSL1( const RGB : TColor32; out H, S, L : Single);
15567: Procedure RGBToHSV( r, g, b : Integer; var h, s, v : Integer)
15568: Procedure RleCompress2( Stream : TStream)
15569: Procedure RleDecompress2( Stream : TStream)
15570: Procedure RmDir(const S: string)
15571: Procedure Rollback
15572: Procedure Rollback( TransDesc : TTransactionDesc)
15573: Procedure RollbackFreeAndNil( var Transaction : TDBXTransaction)
15574: Procedure RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction)
15575: Procedure RollbackTrans
15576: procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer);
15577: Procedure RoundToAllocGranularity64( var Value : Int64; Up : Boolean)
15578: Procedure RoundToAllocGranularityPtr( var Value : Pointer; Up : Boolean)
15579: Procedure RunDll32Internal( Wnd : HWnd; const DLLName, FuncName, CmdLine : string; CmdShow : Integer)
15580: Procedure S_AddMessageToStrings( AMessages : TStrings; AMsg : string)
15581: Procedure S_EBox( const AText : string)
15582: Procedure S_GetEncryptionKeys(DateTime1,DateTime2:TDateTime;var StartKey:int;var MultKey:integer;var
       AddKey:int
15583: Procedure S_IBox( const AText : string)
15584: Procedure S_ReplaceChar( var cStr : string; cOldChr, cNewChr : char)
15585: Procedure S_ReplaceStringInFile( AFileName : string; ASearchString, AReplaceString : string)
15586: Procedure S_TokenInit( cBuffer : PChar; const cDelimiters : string)
15587: Procedure SampleVarianceAndMean
15588: ( const X : TDynFloatArray; var Variance, Mean : Float)
15589: Procedure Save2Click( Sender : TObject)
15590: Procedure Saveas3Click( Sender : TObject)
15591: Procedure Savebefore1Click( Sender : TObject)
15592: Procedure SaveBytesToFile(const Data: TBytes; const FileName: string);
15593: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
15594: Procedure SaveConfigFile
15595: Procedure SaveOutput1Click( Sender : TObject)
15596: procedure SaveScreenshotClick(Sender: TObject);
15597: Procedure SaveLn(pathname, content: string);   //Saveln(exepath+'mysavelntest.txt', memo2.text);
15598: Procedure SaveToClipboardFormat( var AFormat : Word; var AData : THandle; var APalette : HPALETTE)
15599: Procedure SaveToClipboardFormat( var Format : Word; var Data : THandle; var APalette : HPALETTE)
15600: Procedure SaveToFile( AFileName : string)
15601: Procedure SAVETOFILE( const FILENAME : String)
15602: Procedure SaveToFile( const FileName : WideString)
15603: Procedure SaveToFile( const FileName : WideString; Format : TPersistFormat)
15604: Procedure SaveToFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
15605: procedure SaveToFile(FileName: string);
15606: procedure SaveToFile(FileName:String)
15607: Procedure SaveToStream( AStream : TStream; const AHeadersOnly : Boolean)
15608: Procedure SaveToStream( OutStream : TSeekableStream; const Ext : string; Bitmap : TLinarBitmap)
15609: Procedure SaveToStream( S : TStream)
15610: Procedure SaveToStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
15611: Procedure SaveToStream( Stream : TStream)
```

```
15612: Procedure SaveToStream( Stream : TStream; Format : TDataPacketFormat)
15613: procedure SaveToStream(Stream: TStream);
15614: procedure SaveToStream(Stream:TStream)
15615: Procedure SaveToStream1( Stream : TSeekableStream; const FormatExt : string);
15616: Procedure SaveToStream2( Stream : TStream; const FormatExt : string);
15617: Procedure SaveToStrings( AStrings : TStrings; const MimeSeparator : Char)
15618: procedure Say(const sText: string)
15619: Procedure SBytecode1Click( Sender : TObject)
15620: Procedure ScaleImage( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double)
15621: procedure ScriptExplorer1Click(Sender: TObject);
15622: Procedure Scroll( Distance : Integer)
15623: Procedure Scroll( DX, DY : Integer)
15624: procedure ScrollBy(DeltaX, DeltaY: Integer);
15625: procedure SCROLLINVIEW(ACONTROL:TCONTROL)
15626: Procedure ScrollTabs( Delta : Integer)
15627: Procedure Search1Click( Sender : TObject)
15628: procedure SearchAndOpenDoc(vfilenamepath: string)
15629: procedure SearchAndOpenFile(vfilenamepath: string)
15630: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string)
15631: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
15632: Procedure SearchNext1Click( Sender : TObject)
15633: Procedure Select( Node : TTreeNode; ShiftState : TShiftState);
15634: Procedure Select1( const Nodes : array of TTreeNode);
15635: Procedure Select2( Nodes : TList);
15636: Procedure SelectNext( Direction : Boolean)
15637: Procedure SelectNextPage( GoForward : Boolean; CheckTabVisible : Boolean)
15638: Procedure SelfTestPEM  //unit uPSI_cPEM
15639: Procedure Send( AMsg : TIdMessage)
15640: //config forst in const MAILINIFILE = 'maildef.ini';
15641: //ex.:  SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox','
15642: Procedure SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
15643: Procedure SendMail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
15644: Procedure SendMsg( AMsg : TIdMessage; const AHeadersOnly : Boolean)
15645: Procedure SendMsg(AMsg: TIdMessage; const AHeadersOnly: Boolean = False)
15646: Procedure SendResponse
15647: Procedure SendStream( AStream : TStream)
15648: Procedure Set8087CW( NewCW : Word)
15649: Procedure SetAll( One, Two, Three, Four : Byte)
15650: Procedure SetAltRecBuffers( Old, New, Cur : PChar)
15651: Procedure SetAppDispatcher( const ADispatcher : TComponent)
15652: procedure SetArrayLength;
15653: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);  //2 dimension
15654: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
15655: Procedure SetAsHandle( Format : Word; Value : THandle)
15656: procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer)
15657: procedure SetCaptureControl(Control: TControl);
15658: Procedure SetColumnAttributes
15659: Procedure SetCookieField(Values:TStrings;const ADomain,APath:string;AExpires:TDateTime;ASecure:Boolean)
15660: Procedure SetCustomHeader( const Name, Value : string)
15661: Procedure SetExprParams(const Text:Widestring;Opts:TFilterOpts;ParserOpts:TParserOpts;const
       FieldName:Widestring)
15662: Procedure SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
15663: Procedure SetFocus
15664: procedure SetFocus; virtual;
15665: Procedure SetInitialState
15666: Procedure SetKey
15667: procedure SetLastError(ErrorCode: Integer)
15668: procedure SetLength;
15669: Procedure SetLineBreakStyle( var T : Text; Style : TTextLineBreakStyle)
15670: Procedure SETOLE2MENUHANDLE( HANDLE : HMENU)
15671: Procedure SetParams( ADataset : TDataset; UpdateKind : TUpdateKind);
15672: procedure SETPARAMS(APOSITION,AMIN,AMAX:INTEGER)
15673: Procedure SetParams1( UpdateKind : TUpdateKind);
15674: Procedure SetPassword( const Password : string)
15675: Procedure SetPointer( Ptr : Pointer; Size : Longint)
15676: Procedure SetPrimalityTest( const Method : TPrimalityTestMethod)
15677: Procedure SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
15678: Procedure SetProvider( Provider : TComponent)
15679: Procedure SetProxy( const Proxy : string)
15680: Procedure SetPSResult( var PSResult : TPSResult; Value : TObject)
15681: Procedure SetRange( const StartValues, EndValues : array of const)
15682: Procedure SetRangeEnd
15683: Procedure SetRate( const aPercent, aYear : integer)
15684: procedure SetRate(const aPercent, aYear: integer)
15685: Procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
15686: Procedure SetSafeCallExceptionMsg( Msg : String)
15687: procedure SETSELTEXTBUF(BUFFER:PCHAR)
15688: Procedure SetSize( AWidth, AHeight : Integer)
15689: procedure SetSize(NewSize:LongInt)
15690: procedure SetString(var s: string; buffer: PChar; len: Integer)
15691: Procedure SetStrings( List : TStrings)
15692: Procedure SetText( Text : PwideChar)
15693: procedure SetText(Text: PChar);
15694: Procedure SetTextBuf( Buffer : PChar)
15695: procedure SETTEXTBUF(BUFFER:PCHAR)
15696: Procedure SetTick( Value : Integer)
15697: Procedure SetTimeout( ATimeOut : Integer)
15698: Procedure SetTraceEvent( Event : TDBXTraceEvent)
15699: Procedure SetUserName( const UserName : string)
```

```
15700: Procedure SetWallpaper( Path : string);
15701: procedure ShellStyle1Click(Sender: TObject);
15702: Procedure SHORTCUTTOKEY( SHORTCUT : TSHORTCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
15703: Procedure ShowFileProperties( const FileName : string)
15704: Procedure ShowInclude1Click( Sender : TObject)
15705: Procedure ShowInterfaces1Click( Sender : TObject)
15706: Procedure ShowLastException1Click( Sender : TObject)
15707: Procedure ShowMessage( const Msg : string)
15708: Procedure ShowMessageBig(const aText : string);
15709: Procedure ShowMessageBig2(const aText : string; aautosize: boolean);
15710: Procedure ShowMessageBig3(const aText : string; fsize: byte; aautosize: boolean);
15711: Procedure MsgBig(const aText : string);        //alias
15712: procedure showmessage(mytext: string);
15713: Procedure ShowMessageFmt( const Msg : string; Params : array of const)
15714: procedure ShowMessageFmt(const Msg: string; Params: array of const))
15715: Procedure ShowMessagePos( const Msg : string; X, Y : Integer)
15716: procedure ShowMessagePos(const Msg: string; X: Integer; Y: Integer))
15717: Procedure ShowSearchDialog( const Directory : string)
15718: Procedure ShowSpecChars1Click( Sender : TObject)
15719: Procedure ShowBitmap(bmap: TBitmap);  //draw in a form!
15720: Procedure ShredFile( const FileName : string; Times : Integer)
15721: procedure Shuffle(vQ: TStringList);
15722: Procedure ShuffleList( var List : array of Integer; Count : Integer)
15723: Procedure SimulateKeystroke( Key : byte; Shift : TShiftState)
15724: Procedure SinCos( const Theta : Extended; var Sin, Cos : Extended)
15725: Procedure SinCosE( X : Extended; out Sin, Cos : Extended)
15726: Procedure Site( const ACommand : string)
15727: Procedure SkipEOL
15728: Procedure Sleep( ATime : cardinal)
15729: Procedure Sleep( milliseconds : Cardinal)
15730: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
15731: Procedure Slinenumbers1Click( Sender : TObject)
15732: Procedure Sort
15733: Procedure SortColorArray(ColorArray:TColorArray;L,R:Int;SortType:TColorArraySortType;Reverse:Bool)
15734: procedure Speak(const sText: string)   //async like voice
15735: procedure Speak2(const sText: string)  //sync
15736: procedure Split(Str: string;  SubStr: string; List: TStrings);
15737: Procedure SplitNameValue( const Line : string; var Name, Value : string)
15738: Procedure SplitColumns( const AData : String; AStrings : TStrings; const ADelim : String)
15739: Procedure SplitColumnsNoTrim( const AData : String; AStrings : TStrings; const ADelim : String)
15740: Procedure SplitLines( AData : PChar; ADataSize : Integer; AStrings : TStrings)
15741: Procedure SplitString( const AStr, AToken : String; var VLeft, VRight : String)
15742: procedure SQLSyntax1Click(Sender: TObject);
15743: Procedure SRand( Seed : RNG_IntType)
15744: Procedure Start
15745: Procedure StartCount( var Counter : TJclCounter; const Compensate : Boolean)
15746: procedure StartFileFinder3(spath,aext,searchstr: string; arecursiv: boolean; reslist: TStringlist);
15747: Procedure StartTransaction( TransDesc : TTransactionDesc)
15748: Procedure Status( var AStatusList : TStringList)
15749: Procedure StatusBar1DblClick( Sender : TObject)
15750: Procedure StepInto1Click( Sender : TObject)
15751: Procedure StepIt
15752: Procedure StepOut1Click( Sender : TObject)
15753: Procedure Stop
15754: procedure stopmp3;
15755: procedure StartWeb(aurl: string);
15756: Procedure Str(aint: integer; astr: string); //of system
15757: Procedure StrDispose( Str : PChar)
15758: procedure StrDispose(Str: PChar)
15759: Procedure StrReplace(var Str: String; Old, New: String);
15760: Procedure StretchDIBits( DC : THandle; const Dest : TRect; HalftoneStretch : Boolean)
15761: procedure StretchDraw(const Rect: TRect; Graphic: TGraphic);
15762: Procedure StringToBytes( Value : String; Bytes : array of byte)
15763: procedure StrSet(c : Char; I : Integer; var s : String);
15764: Procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
15765: Procedure StructureMount( APath : String)
15766: procedure STYLECHANGED(SENDER:TOBJECT)
15767: Procedure Subselect( Node : TTreeNode; Validate : Boolean)
15768: procedure Succ(X: int64);
15769: Procedure SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares : Extended)
15770: procedure SwapChar(var X,Y: char);  //swapX follows
15771: Procedure SwapFloats( var X, Y : Float)
15772: procedure SwapGrid(grd: TStringGrid);
15773: Procedure SwapOrd( var I, J : Byte);
15774: Procedure SwapOrd( var X, Y : Integer)
15775: Procedure SwapOrd1( var I, J : Shortint);
15776: Procedure SwapOrd2( var I, J : Smallint);
15777: Procedure SwapOrd3( var I, J : Word);
15778: Procedure SwapOrd4( var I, J : Integer);
15779: Procedure SwapOrd5( var I, J : Cardinal);
15780: Procedure SwapOrd6( var I, J : Int64);
15781: Procedure SymetricCompareFiles(const plaintext, replaintext: string)
15782: Procedure Synchronize1( Method : TMethod);
15783: procedure SyntaxCheck1Click(Sender: TObject);
15784: Procedure SysFreeString(const S: WideString); stdcall;
15785: Procedure TakeOver( Other : TLinearBitmap)
15786: Procedure Talkln(const sText: string)  //async voice
15787: procedure tbtn6resClick(Sender: TObject);
15788: Procedure tbtnUseCaseClick( Sender : TObject)
```

```
15789: procedure TerminalStyle1Click(Sender: TObject);
15790: Procedure Terminate
15791: Procedure texSyntax1Click( Sender : TObject)
15792: procedure TextOut(X, Y: Integer; Text: string);
15793: Procedure TextRect( Rect : TRect; X, Y : Integer; const Text : string);
15794: procedure TextRect(Rect: TRect; X: Integer; Y: Integer; const Text: string);
15795: Procedure TextRect1( var Rect : TRect; var Text : string; TextFormat : TTextFormat);
15796: Procedure TextStart
15797: procedure TILE
15798: Procedure TimeStampToBytes( Value : TBcd; Bytes : array of byte)
15799: Procedure TitleClick( Column : TColumn)
15800: Procedure ToDo
15801: procedure toolbtnTutorialClick(Sender: TObject);
15802: Procedure Trace1( AURL : string; const AResponseContent : TStream);
15803: Procedure TransferMode( ATransferMode : TIdFTPTransferMode)
15804: Procedure Truncate
15805: procedure Tutorial101Click(Sender: TObject);
15806: procedure Tutorial10Statistics1Click(Sender: TObject);
15807: procedure Tutorial11Forms1Click(Sender: TObject);
15808: procedure Tutorial12SQL1Click(Sender: TObject);
15809: Procedure tutorial1Click( Sender : TObject)
15810: Procedure tutorial21Click( Sender : TObject)
15811: Procedure tutorial31Click( Sender : TObject)
15812: Procedure tutorial4Click( Sender : TObject)
15813: Procedure Tutorial5Click( Sender : TObject)
15814: procedure Tutorial6Click(Sender: TObject);
15815: procedure Tutorial91Click(Sender: TObject);
15816: Procedure UnhookSignal( RtlSigNum : Integer; OnlyIfHooked : Boolean)
15817: procedure UniqueString(var str: AnsiString)
15818: procedure UnloadLoadPackage(Module: HMODULE)
15819: Procedure Unlock
15820: Procedure UNMERGE( MENU : TMAINMENU)
15821: Procedure UnRegisterChanges( Value : TChangeLink)
15822: Procedure UnregisterConversionFamily( const AFamily : TConvFamily)
15823: Procedure UnregisterConversionType( const AType : TConvType)
15824: Procedure UnRegisterProvider( Prov : TCustomProvider)
15825: Procedure UPDATE
15826: Procedure UpdateBatch( AffectRecords : TAffectRecords)
15827: Procedure UPDATECURSORPOS
15828: Procedure UpdateFile
15829: Procedure UpdateItems( FirstIndex, LastIndex : Integer)
15830: Procedure UpdateResponse( AResponse : TWebResponse)
15831: Procedure UpdateScrollBar
15832: Procedure UpdateView1Click( Sender : TObject)
15833: procedure Val(const s: string; var n, z: Integer)
15834: procedure VarArraySet(c : Variant; I : Integer; var s : Variant);
15835: Procedure VarFMTBcdCreate( var ADest : Variant; const ABcd : TBcd);
15836: Procedure VariantAdd( const src : Variant; var dst : Variant)
15837: Procedure VariantAnd( const src : Variant; var dst : Variant)
15838: Procedure VariantArrayRedim( var V : Variant; High : Integer)
15839: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
15840: Procedure VariantClear( var V : Variant)
15841: Procedure VariantCpy( const src : Variant; var dst : Variant)
15842: Procedure VariantDiv( const src : Variant; var dst : Variant)
15843: Procedure VariantMod( const src : Variant; var dst : Variant)
15844: Procedure VariantMul( const src : Variant; var dst : Variant)
15845: Procedure VariantOr( const src : Variant; var dst : Variant)
15846: Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
15847: Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
15848: Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
15849: Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
15850: Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
15851: Procedure VariantShl( const src : Variant; var dst : Variant)
15852: Procedure VariantShr( const src : Variant; var dst : Variant)
15853: Procedure VariantSub( const src : Variant; var dst : Variant)
15854: Procedure VariantXor( const src : Variant; var dst : Variant)
15855: Procedure VarCastError;
15856: Procedure VarCastError1( const ASourceType, ADestType : TVarType);
15857: Procedure VarInvalidOp
15858: Procedure VarInvalidNullOp
15859: Procedure VarOverflowError( const ASourceType, ADestType : TVarType)
15860: Procedure VarRangeCheckError( const ASourceType, ADestType : TVarType)
15861: Procedure VarArrayCreateError
15862: Procedure VarResultCheck( AResult : HRESULT);
15863: Procedure VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);
15864: Procedure HandleConversionException( const ASourceType, ADestType : TVarType)
15865: Function VarTypeAsText( const AType : TVarType) : string
15866: procedure Voice(const sText: string) //async
15867: procedure Voice2(const sText: string) //sync
15868: Procedure WaitMiliSeconds( AMSec : word)
15869: Procedure WideAppend( var dst : WideString; const src : WideString)
15870: Procedure WideAssign( var dst : WideString; var src : WideString)
15871: Procedure WideDelete( var dst : WideString; index, count : Integer)
15872: Procedure WideFree( var s : WideString)
15873: Procedure WideFromAnsi( var dst : WideString; const src : AnsiString)
15874: Procedure WideFromPChar( var dst : WideString; src : PChar)
15875: Procedure WideInsert( var dst : WideString; const src : WideString; index : Integer)
15876: Procedure WideSetLength( var dst : WideString; len : Integer)
15877: Procedure WideString2Stream( aWideString : WideString; oStream : TStream)
```

```
15878: Procedure WideStringToBytes( Value : WideString; Bytes : array of byte)
15879: Procedure WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
15880: Procedure WinInet_HttpGet(const Url: string; Stream:TStream);
15881: Procedure HttpGet(const Url: string; Stream:TStream);
15882: Procedure WordToTwoBytes( AWord : Word; ByteArray : TIdBytes; Index : integer)
15883: Procedure WordWrap1Click( Sender : TObject)
15884: Procedure Write( const AOut : string)
15885: Procedure Write( Socket : TSocket)
15886: procedure Write(S: string);
15887: Procedure WriteBinaryStream( const Section, Name : string; Value : TStream)
15888: Procedure WriteBool( const Section, Ident : string; Value : Boolean)
15889: Procedure WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow : Boolean)
15890: procedure WriteBuffer(Buffer:String;Count:LongInt)
15891: Procedure WriteCardinal( AValue : Cardinal; const AConvert : Boolean)
15892: Procedure WriteChar( AValue : Char)
15893: Procedure WriteDate( const Section, Name : string; Value : TDateTime)
15894: Procedure WriteDateTime( const Section, Name : string; Value : TDateTime)
15895: Procedure WriteFloat( const Section, Name : string; Value : Double)
15896: Procedure WriteHeader( AHeader : TStrings)
15897: Procedure WriteInteger( AValue : Integer; const AConvert : Boolean)
15898: Procedure WriteInteger( const Section, Ident : string; Value : Longint)
15899: Procedure WriteLn( const AOut : string)
15900: procedure Writeln(s: string);
15901: Procedure WriteLog( const FileName, LogLine : string)
15902: Procedure WriteRFCReply( AReply : TIdRFCReply)
15903: Procedure WriteRFCStrings( AStrings : TStrings)
15904: Procedure WriteSmallInt( AValue : SmallInt; const AConvert : Boolean)
15905: Procedure WriteStream(AStream:TStream;const AAll:Bool;const AWriteByteCount:Bool;const ASize:Int)
15906: Procedure WriteString( const Section, Ident, Value : String)
15907: Procedure WriteStrings( AValue : TStrings; const AWriteLinesCount : Boolean)
15908: Procedure WriteTime( const Section, Name : string; Value : TDateTime)
15909: Procedure WriteObjectResourceHeader( ObjStream, Output : TStream)
15910: Procedure Write16bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
15911: Procedure Write32bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
15912: procedure WStrSet(c : AnyString; I : Integer; var s : AnyString);
15913: procedure XMLSyntax1Click(Sender: TObject);
15914: Procedure XOR_Streams2( Dest, Srce : TMemoryStream)
15915: Procedure XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
15916: Procedure ZeroFillStream( Stream : TMemoryStream)
15917: procedure XMLSyntax1Click(Sender: TObject);
15918: Procedure ZeroMemory( Ptr : Pointer; Length : Longint)
15919: procedure(Control: TWinControl; Index: Integer; Rect: TRect; State: Byte)
15920: procedure(Control: TWinControl; Index: Integer; var Height: Integer)
15921: procedure(Sender, Source: TObject; X, Y: Integer; State: TDragState; var Accept: Boolean)
15922: procedure(Sender, Source: TObject;X, Y: Integer)
15923: procedure(Sender, Target: TObject; X, Y: Integer)
15924: procedure(Sender: TObject; ASection, AWidth: Integer)
15925: procedure(Sender: TObject; ScrollCode: TScrollCode;var ScrollPos: Integer)
15926: procedure(Sender: TObject; Shift: TShiftState; X, Y: Integer);
15927: procedure(Sender: TObject; var Action: TCloseAction)
15928: procedure(Sender: TObject; var CanClose: Boolean)
15929: procedure(Sender: TObject; var Key: Char);
15930: ProcedureName ProcedureNames ProcedureParametersCursor @
15931:
15932: *************Now Constructors constructor *************
15933:  Size is: 1115 996 628 550 544 501 459 (381) (360) (270 246) (235)
15934:  Attach( VersionInfoData : Pointer; Size : Integer)
15935:  constructor Create( ABuckets : TBucketListSizes)
15936:  Create( ACallBackWnd : HWND)
15937:  Create( AClient : TCustomTaskDialog)
15938:  Create( AClient : TIdTelnet)
15939:  Create( ACollection : TCollection)
15940:  Create( ACollection : TFavoriteLinkItems)
15941:  Create( ACollection : TTaskDialogButtons)
15942:  Create( AConnection : TIdCustomHTTP)
15943:  Create( ACreateSuspended : Boolean)
15944:  Create( ADataSet : TCustomSQLDataSet)
15945:  CREATE( ADATASET : TDATASET)
15946:  Create( Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
15947:  Create( AGrid : TCustomDBGrid)
15948:  Create( AGrid : TStringGrid; AIndex : Longint)
15949:  Create( AHTTP : TIdCustomHTTP)
15950:  Create( AListItems : TListItems)
15951:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
15952:  Create( AOnBytesRemoved : TIdBufferBytesRemoved)
15953:  Create( AOwner : TCommonCalendar)
15954:  Create( AOwner : TComponent)
15955:  CREATE( AOWNER : TCOMPONENT)
15956:  Create( AOwner : TCustomListView)
15957:  Create( AOwner : TCustomOutline)
15958:  Create( AOwner : TCustomRichEdit)
15959:  Create( AOwner : TCustomRichEdit; AttributeType : TAttributeType)
15960:  Create( AOwner : TCustomTreeView)
15961:  Create( AOwner : TIdUserManager)
15962:  Create( AOwner : TListItems)
15963:  Create( AOwner :
       TObject;Handle:hDBICur;CBType:CBType;CBBuf:Pointer;CBBufSize:Int;CallbackEvent:TBDECallbackEvent; Chain :
       Boolean)
15964:  CREATE( AOWNER : TPERSISTENT)
```

```
15965:  Create( AOwner : TPersistent)
15966:  Create( AOwner : TTable)
15967:  Create( AOwner : TTreeNodes)
15968:  Create( AOwner : TWinControl; const ClassName : string)
15969:  Create( AParent : TIdCustomHTTP)
15970:  Create( AParent : TUpdateTree; AResolver : TCustomResolver)
15971:  Create( AProvider : TBaseProvider)
15972:  Create( AProvider : TCustomProvider);
15973:  Create( AProvider : TDataSetProvider)
15974:  Create( ASocket : TCustomWinSocket; TimeOut : Longint)
15975:  Create( ASocket : TSocket)
15976:  Create( AStrings : TWideStrings)
15977:  Create( AToolBar : TToolBar)
15978:  Create( ATreeNodes : TTreeNodes)
15979:  Create( Autofill : boolean)
15980:  Create( AWebPageInfo : TAbstractWebPageInfo)
15981:  Create( AWebRequest : TWebRequest)
15982:  Create( Collection : TCollection)
15983:  Create( Collection : TIdMessageParts; ABody : TStrings)
15984:  Create( Collection : TIdMessageParts; const AFileName : TFileName)
15985:  Create( Column : TColumn)
15986:  Create( const AConvFamily : TConvFamily; const ADescription : string)
15987:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AFactor : Double)
15988:  Create( const AConvFamily : TConvFamily; const ADescription : string; const AToCommonProc,
       AFromCommonProc : TConversionProc)
15989:  Create( const AInitialState : Boolean; const AManualReset : Boolean)
15990:  Create( const ATabSet : TTabSet)
15991:  Create( const Compensate : Boolean)
15992:  Create( const FileMap : TJclCustomFileMapping; Access, Size : Cardinal; ViewOffset : Int64)
15993:  Create( const FileName : string)
15994:  Create( const FileName : string;FileMode:Cardinal; const Name:string; Protect:Cardinal; const MaximumSize
       : Int64; const SecAttr : PSecurityAttributes);
15995:  Create( const FileName : string; FileMode : WordfmShareDenyWrite)
15996:  Create( const MaskValue : string)
15997:  Create(const Name:string; Protect:Cardinal;const MaximumSize:Int64;const SecAttr:PSecurityAttributes)
15998:  Create( const Prefix : string)
15999:  Create( const sRegularExpression : string; xFlags : TniRegularExpressionMatchFlags)
16000:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
16001:  Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
16002:  Create( CoolBar : TCoolBar)
16003:  Create( CreateSuspended : Boolean; ASocket : TServerClientWinSocket)
16004:  Create( CreateSuspended : Boolean; ASocket : TServerWinSocket)
16005:  Create( DataSet :TDataSet; const
       Text:Widestring;Options:TFilterOptions;ParserOptions:TParserOptions;const FieldName : Widestring;
       DepFields : TBits; FieldMap : TFieldMap)
16006:  Create( DBCtrlGrid : TDBCtrlGrid)
16007:  Create( DSTableProducer : TDSTableProducer)
16008:  Create( DSTableProducer : TDSTableProducer; ColumnClass : THTMLTableColumnClass)
16009:  Create( ErrorCode : DBIResult)
16010:  Create( Field : TBlobField; Mode : TBlobStreamMode)
16011:  Create( Grid : TCustomDBGrid; ColumnClass : TColumnClass)
16012:  Create( HeaderControl : TCustomHeaderControl)
16013:  Create( HTTPRequest : TWebRequest)
16014:  Create( iStart : integer; sText : string)
16015:  Create( iValue : Integer)
16016:  Create( Kind : TMmTimerKind; Notification : TMmNotificationKind)
16017:  Create( MciErrNo : MCIERROR; const Msg : string)
16018:  Create(MemoryStream:TCustomMemStream;FreeStream:Boolean;const AIndexOption:TJclMappedTextReaderIndex);
16019:  Create( Message : string; ErrorCode : DBResult)
16020:  Create( Msg : string)
16021:  Create( NativeError, Context : string; ErrCode, PrevError : Integer; E : Exception)
16022:  Create( NativeError, Context : string; ErrorCode, PreviousError : DBResult)
16023:  Create( oExpression : TniRegularExpression; eType : TniRegularExpressionStateType)
16024:  Create( oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags)
16025:  Create(oSource:TniRegularExpressState;oDestination:TniRegularExprState;xCharacts:TCharSet;bLambda:bool)
16026:  Create( Owner : EDBEngineError; ErrorCode : DBIResult; NativeError : Longint; Message : PChar)
16027:  Create( Owner : TCustomComboBoxEx)
16028:  CREATE( OWNER : TINDEXDEFS; const NAME, FIELDS: String; OPTIONS: TINDEXOPTIONS)
16029:  Create( Owner : TPersistent)
16030:  Create( Params : TStrings)
16031:  Create( Size : Cardinal)
16032:  Create( Socket : TSocket; ServerWinSocket : TServerWinSocket)
16033:  Create( StatusBar : TCustomStatusBar)
16034:  Create( WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass)
16035:  Create( WebResponse : TWebResponse; ItemClass : TCollectionItemClass)
16036:  Create(AHandle:Integer)
16037:  Create(AOwner: TComponent); virtual;
16038:  Create(const AURI : string)
16039:  Create(FileName:String;Mode:Word)
16040:  Create(Instance:THandle;ResName:String;ResType:PChar)
16041:  Create(Stream : TStream)
16042:  Create1( ADataset : TDataset);
16043:  Create1(const FileHandle:THandle;const Name:string;Protect:Cardinal;const MaximumSize:Int64;const
       SecAttr:PSecurityAttributes);
16044:  Create1( const FileName : string; const AIndexOption : TJclMappedTextReaderIndex);
16045:  Create2( Other : TObject);
16046:  CreateAt(FileMap: TJclCustomFileMapping; Access,Size:Cardinal;ViewOffset:Int64;Address: Pointer)
16047:  CreateError(const anErrCode: Integer;const asReplyMessage: string; const asErrorMessage: string)
16048:  CreateFmt( MciErrNo : MCIERROR; const Msg : string; const Args : array of const)
```

```
16049:  CreateFromId(Instance:THandle;ResId:Integer;ResType:PChar)
16050:  CreateLinked( DBCtrlGrid : TDBCtrlGrid)
16051:  CREATENEW(AOWNER:TCOMPONENT; Dummy: Integer)
16052:  CreateRes( Ident : Integer);
16053:  CreateRes( MciErrNo : MCIERROR; Ident : Integer)
16054:  CreateRes( ResStringRec : PResStringRec);
16055:  CreateResHelp( Ident : Integer; AHelpContext : Integer);
16056:  CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Integer);
16057:  CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
16058:  CreateSize( AWidth, AHeight : Integer)
16059:  Open( const Name : string; const InheritHandle : Boolean; const DesiredAccess : Cardinal)
16060:
16061: ------------------------------------------------------------------------------
16062: unit uPSI_MathMax;
16063: ------------------------------------------------------------------------------
16064:  CONSTS
16065:   Bernstein: Float = 0.280169499023869133036436491230 7;  // Bernstein constant
16066:   Cbrt2: Float     = 1.259921049894873164767210607278 2;  // CubeRoot(2)
16067:   Cbrt3: Float     = 1.442249570307408382321638310780 1;  // CubeRoot(3)
16068:   Cbrt10: Float    = 2.154434690031883721759293566519 4;  // CubeRoot(10)
16069:   Cbrt100: Float   = 4.641588833612778892410076350919 4;  // CubeRoot(100)
16070:   CbrtPi: Float    = 1.464591887561523263020142527263 8;  // CubeRoot(PI)
16071:   Catalan: Float   = 0.915965594177219015054603514932 4;  // Catalan constant
16072:   PiJ:   Float       = 3.141592653589793238462643383279 5;  // PI
16073:   PI:  Extended =  3.141592653589793238462643383279 5);
16074:   PiOn2: Float     = 1.570796326794896619231321691639 8;  // PI / 2
16075:   PiOn3: Float     = 1.047197551196597746154214461093 2;  // PI / 3
16076:   PiOn4: Float     = 0.785398163397448309615660845819 88; // PI / 4
16077:   Sqrt2: Float     = 1.414213562373095048801688724209 7;  // Sqrt(2)
16078:   Sqrt3: Float     = 1.732050807568877293527446341505 9;  // Sqrt(3)
16079:   Sqrt5: Float     = 2.236067977499789696409173668731 3;  // Sqrt(5)
16080:   Sqrt10: Float    = 3.162277660168379331998893544432 7;  // Sqrt(10)
16081:   SqrtPi: Float    = 1.772453850905516027298167483341 1;  // Sqrt(PI)
16082:   Sqrt2Pi: Float   = 2.506628274631000502415765284811;   // Sqrt(2 * PI)
16083:   TwoPi: Float     = 6.283185307179586476925286766559;   // 2 * PI
16084:   ThreePi: Float   = 9.424777960769379715387930149838 5;  // 3 * PI
16085:   Ln2: Float       = 0.693147180559945309417232121458 18; // Ln(2)
16086:   Ln10: Float      = 2.302585092994045684017991454684 4;  // Ln(10)
16087:   LnPi: Float      = 1.144729885849400174143427351353 1;  // Ln(PI)
16088:   Log2J: Float     = 0.301029995663981195213738894724 49; // Log10(2)
16089:   Log3: Float      = 0.477121254719662437295027903255 12; // Log10(3)
16090:   LogPi: Float     = 0.497149872694133854351268288290 9;  // Log10(PI)
16091:   LogE: Float      = 0.434294481903251827651128918916 61; // Log10(E)
16092:   E: Float         = 2.718281828459045235360287471352 7;  // Natural constant
16093:   hLn2Pi: Float    = 0.918938533204672741780329736405 62; // Ln(2*PI)/2
16094:   inv2Pi: Float    = 0.159154943091895335768883763372514362034459645740 46; // 0.5/Pi
16095:   TwoToPower63: Float = 9223372036854775808.0;           // 2^63
16096:   GoldenMean: Float   = 1.618033988749894848204586834365638;  // GoldenMean
16097:   EulerMascheroni: Float = 0.577215664901532860606512090082 4;  // Euler GAMMA
16098:   RadCor : Double = 57.29577951308232;    {number of degrees in a radian}
16099:   StDelta         : Extended = 0.00001;  {delta for difference equations}
16100:   StEpsilon       : Extended = 0.00001;  {epsilon for difference equations}
16101:   StMaxIterations : Integer  = 100;      {max attempts for convergence}
16102:
16103:   MaxAngle',( 9223372036854775808.0);
16104:   MaxTanH',( 5678.261703147071974745965538985 4);
16105:   MaxFactorial','LongInt').SetInt( 1754);
16106:   MaxFloatingPoint',(1.189731495357231765085759326628E+4932);
16107:   MinFloatingPoint',(3.362103143112093506262677817321 8E-4932);
16108:   MaxTanH',( 354.891356446691998421622846186 59);
16109:   MaxFactorial','LongInt').SetInt( 170);
16110:   MaxFloatingPointD',(1.797693134862315907729305190789E+308);
16111:   MinFloatingPointD',(2.225073858507201383090232717332 4E-308);
16112:   MaxTanH',( 44.361419555836499802702855773323);
16113:   MaxFactorial','LongInt').SetInt( 33);
16114:   MaxFloatingPointS',( 3.402823669209384634633746074317 7E+38);
16115:   MinFloatingPointS',( 1.175494350822287507968736537222 2E-38);
16116:   PiExt',( 3.141592653589793238462643383279 5);
16117:   RatioDegToRad',( PiExt / 180.0);
16118:   RatioGradToRad',( PiExt / 200.0);
16119:   RatioDegToGrad',( 200.0 / 180.0);
16120:   RatioGradToDeg',( 180.0 / 200.0);
16121:   Crc16PolynomCCITT','LongWord').SetUInt( $1021);
16122:  Crc16PolynomIBM','LongWord').SetUInt( $8005);
16123:  Crc16Bits','LongInt').SetInt( 16);
16124:  Crc16Bytes','LongInt').SetInt( 2);
16125:  Crc16HighBit','LongWord').SetUInt( $8000);
16126:  NotCrc16HighBit','LongWord').SetUInt( $7FFF);
16127:   Crc32PolynomIEEE','LongWord').SetUInt( $04C11DB7);
16128:  Crc32PolynomCastagnoli','LongWord').SetUInt( $1EDC6F41);
16129:  Crc32Koopman','LongWord').SetUInt( $741B8CD7);
16130:  Crc32Bits','LongInt').SetInt( 32);
16131:  Crc32Bytes','LongInt').SetInt( 4);
16132:  Crc32HighBit','LongWord').SetUInt( $80000000);
16133:  NotCrc32HighBit','LongWord').SetUInt( $7FFFFFFF);
16134:
16135:   MinByte       = Low(Byte);
16136:   MaxByte       = High(Byte);
16137:   MinWord       = Low(Word);
```

```
16138:   MaxWord       = High(Word);
16139:   MinShortInt   = Low(ShortInt);
16140:   MaxShortInt   = High(ShortInt);
16141:   MinSmallInt   = Low(SmallInt);
16142:   MaxSmallInt   = High(SmallInt);
16143:   MinLongWord   = LongWord(Low(LongWord));
16144:   MaxLongWord   = LongWord(High(LongWord));
16145:   MinLongInt    = LongInt(Low(LongInt));
16146:   MaxLongInt    = LongInt(High(LongInt));
16147:   MinInt64      = Int64(Low(Int64));
16148:   MaxInt64      = Int64(High(Int64));
16149:   MinInteger    = Integer(Low(Integer));
16150:   MaxInteger    = Integer(High(Integer));
16151:   MinCardinal   = Cardinal(Low(Cardinal));
16152:   MaxCardinal   = Cardinal(High(Cardinal));
16153:   MinNativeUInt = NativeUInt(Low(NativeUInt));
16154:   MaxNativeUInt = NativeUInt(High(NativeUInt));
16155:   MinNativeInt  = NativeInt(Low(NativeInt));
16156:   MaxNativeInt  = NativeInt(High(NativeInt));
16157:  Function CosH( const Z : Float) : Float;
16158:  Function SinH( const Z : Float) : Float;
16159:  Function TanH( const Z : Float) : Float;
16160:
16161:
16162:   //***********from DMath.Dll Lib of types.inc in source\dmath_dll
16163:   InvLn2     = 1.44269504088896340736;  { 1/Ln(2) }
16164:   InvLn10    = 0.43429448190325182765;  { 1/Ln(10) }
16165:   TwoPi      = 6.28318530717958647693;  { 2*Pi }
16166:   PiDiv2     = 1.57079632679489661923;  { Pi/2 }
16167:   SqrtPi     = 1.77245385090551602730;  { Sqrt(Pi) }
16168:   Sqrt2Pi    = 2.50662827463100050242;  { Sqrt(2*Pi) }
16169:   InvSqrt2Pi = 0.39894228040143267794;  { 1/Sqrt(2*Pi) }
16170:   LnSqrt2Pi  = 0.91893853320467274178;  { Ln(Sqrt(2*Pi)) }
16171:   Ln2PiDiv2  = 0.91893853320467274178;  { Ln(2*Pi)/2 }
16172:   Sqrt2      = 1.41421356237309504880;  { Sqrt(2) }
16173:   Sqrt2Div2  = 0.70710678118654752440;  { Sqrt(2)/2 }
16174:   Gold       = 1.61803398874989484821;  { Golden Mean = (1 + Sqrt(5))/2 }
16175:   CGold      = 0.38196601125010515179;  { 2 - GOLD }
16176:   MachEp = 2.220446049250313E-16;   { 2^(-52) }
16177:   MaxNum = 1.797693134862315E+308;  { 2^1024 }
16178:   MinNum = 2.225073858507202E-308;  { 2^(-1022) }
16179:   MaxLog = 709.7827128933840;
16180:   MinLog = -708.3964185322641;
16181:   MaxFac = 170;
16182:   MaxGam = 171.624376956302;
16183:   MaxLgm = 2.556348E+305;
16184:   SingleCompareDelta   = 1.0E-34;
16185:   DoubleCompareDelta   = 1.0E-280;
16186:   {$IFDEF CLR}
16187:   ExtendedCompareDelta = DoubleCompareDelta;
16188:   {$ELSE}
16189:   ExtendedCompareDelta = 1.0E-4400;
16190:   {$ENDIF}
16191:   Bytes1KB  = 1024;
16192:   Bytes1MB  = 1024 * Bytes1KB;
16193:   Bytes1GB  = 1024 * Bytes1MB;
16194:   Bytes64KB = 64 * Bytes1KB;
16195:   Bytes64MB = 64 * Bytes1MB;
16196:   Bytes2GB  = 2 * LongWord(Bytes1GB);
16197:     clBlack32', $FF000000 ));
16198:     clDimGray32', $FF3F3F3F ));
16199:     clGray32', $FF7F7F7F ));
16200:     clLightGray32', $FFBFBFBF ));
16201:     clWhite32', $FFFFFFFF ));
16202:     clMaroon32', $FF7F0000 ));
16203:     clGreen32', $FF007F00 ));
16204:     clOlive32', $FF7F7F00 ));
16205:     clNavy32', $FF00007F ));
16206:     clPurple32', $FF7F007F ));
16207:     clTeal32', $FF007F7F ));
16208:     clRed32', $FFFF0000 ));
16209:     clLime32', $FF00FF00 ));
16210:     clYellow32', $FFFFFF00 ));
16211:     clBlue32', $FF0000FF ));
16212:     clFuchsia32', $FFFF00FF ));
16213:     clAqua32', $FF00FFFF ));
16214:     clAliceBlue32', $FFF0F8FF ));
16215:     clAntiqueWhite32', $FFFAEBD7 ));
16216:     clAquamarine32', $FF7FFFD4 ));
16217:     clAzure32', $FFF0FFFF ));
16218:     clBeige32', $FFF5F5DC ));
16219:     clBisque32', $FFFFE4C4 ));
16220:     clBlancheDalmond32', $FFFFEBCD ));
16221:     clBlueViolet32', $FF8A2BE2 ));
16222:     clBrown32', $FFA52A2A ));
16223:     clBurlyWood32', $FFDEB887 ));
16224:     clCadetblue32', $FF5F9EA0 ));
16225:     clChartReuse32', $FF7FFF00 ));
16226:     clChocolate32', $FFD2691E ));
```

```
16227:     clCoral32', $FFFF7F50 ));
16228:     clCornFlowerBlue32', $FF6495ED ));
16229:     clCornSilk32', $FFFFF8DC ));
16230:     clCrimson32', $FFDC143C ));
16231:     clDarkBlue32', $FF00008B ));
16232:     clDarkCyan32', $FF008B8B ));
16233:     clDarkGoldenRod32', $FFB8860B ));
16234:     clDarkGray32', $FFA9A9A9 ));
16235:     clDarkGreen32', $FF006400 ));
16236:     clDarkGrey32', $FFA9A9A9 ));
16237:     clDarkKhaki32', $FFBDB76B ));
16238:     clDarkMagenta32', $FF8B008B ));
16239:     clDarkOliveGreen32', $FF556B2F ));
16240:     clDarkOrange32', $FFFF8C00 ));
16241:     clDarkOrchid32', $FF9932CC ));
16242:     clDarkRed32', $FF8B0000 ));
16243:     clDarkSalmon32', $FFE9967A ));
16244:     clDarkSeaGreen32', $FF8FBC8F ));
16245:     clDarkSlateBlue32', $FF483D8B ));
16246:     clDarkSlateGray32', $FF2F4F4F ));
16247:     clDarkSlateGrey32', $FF2F4F4F ));
16248:     clDarkTurquoise32', $FF00CED1 ));
16249:     clDarkViolet32', $FF9400D3 ));
16250:     clDeepPink32', $FFFF1493 ));
16251:     clDeepSkyBlue32', $FF00BFFF ));
16252:     clDodgerBlue32', $FF1E90FF ));
16253:     clFireBrick32', $FFB22222 ));
16254:     clFloralWhite32', $FFFFFAF0 ));
16255:     clGainsBoro32', $FFDCDCDC ));
16256:     clGhostWhite32', $FFF8F8FF ));
16257:     clGold32', $FFFFD700 ));
16258:     clGoldenRod32', $FFDAA520 ));
16259:     clGreenYellow32', $FFADFF2F ));
16260:     clGrey32', $FF808080 ));
16261:     clHoneyDew32', $FFF0FFF0 ));
16262:     clHotPink32', $FFFF69B4 ));
16263:     clIndianRed32', $FFCD5C5C ));
16264:     clIndigo32', $FF4B0082 ));
16265:     clIvory32', $FFFFFFF0 ));
16266:     clKhaki32', $FFF0E68C ));
16267:     clLavender32', $FFE6E6FA ));
16268:     clLavenderBlush32', $FFFFF0F5 ));
16269:     clLawnGreen32', $FF7CFC00 ));
16270:     clLemonChiffon32', $FFFFFACD ));
16271:     clLightBlue32', $FFADD8E6 ));
16272:     clLightCoral32', $FFF08080 ));
16273:     clLightCyan32', $FFE0FFFF ));
16274:     clLightGoldenRodYellow32', $FFFAFAD2 ));
16275:     clLightGreen32', $FF90EE90 ));
16276:     clLightGrey32', $FFD3D3D3 ));
16277:     clLightPink32', $FFFFB6C1 ));
16278:     clLightSalmon32', $FFFFA07A ));
16279:     clLightSeagreen32', $FF20B2AA ));
16280:     clLightSkyblue32', $FF87CEFA ));
16281:     clLightSlategray32', $FF778899 ));
16282:     clLightSlategrey32', $FF778899 ));
16283:     clLightSteelblue32', $FFB0C4DE ));
16284:     clLightYellow32', $FFFFFFE0 ));
16285:     clLtGray32', $FFC0C0C0 ));
16286:     clMedGray32', $FFA0A0A4 ));
16287:     clDkGray32', $FF808080 ));
16288:     clMoneyGreen32', $FFC0DCC0 ));
16289:     clLegacySkyBlue32', $FFA6CAF0 ));
16290:     clCream32', $FFFFFBF0 ));
16291:     clLimeGreen32', $FF32CD32 ));
16292:     clLinen32', $FFFAF0E6 ));
16293:     clMediumAquamarine32', $FF66CDAA ));
16294:     clMediumBlue32', $FF0000CD ));
16295:     clMediumOrchid32', $FFBA55D3 ));
16296:     clMediumPurple32', $FF9370DB ));
16297:     clMediumSeaGreen32', $FF3CB371 ));
16298:     clMediumSlateBlue32', $FF7B68EE ));
16299:     clMediumSpringGreen32', $FF00FA9A ));
16300:     clMediumTurquoise32', $FF48D1CC ));
16301:     clMediumVioletRed32', $FFC71585 ));
16302:     clMidnightBlue32', $FF191970 ));
16303:     clMintCream32', $FFF5FFFA ));
16304:     clMistyRose32', $FFFFE4E1 ));
16305:     clMoccasin32', $FFFFE4B5 ));
16306:     clNavajoWhite32', $FFFFDEAD ));
16307:     clOldLace32', $FFFDF5E6 ));
16308:     clOliveDrab32', $FF6B8E23 ));
16309:     clOrange32', $FFFFA500 ));
16310:     clOrangeRed32', $FFFF4500 ));
16311:     clOrchid32', $FFDA70D6 ));
16312:     clPaleGoldenRod32', $FFEEE8AA ));
16313:     clPaleGreen32', $FF98FB98 ));
16314:     clPaleTurquoise32', $FFAFEEEE ));
16315:     clPaleVioletred32', $FFDB7093 ));
```

```
16316:     clPapayaWhip32', $FFFFEFD5 ));
16317:     clPeachPuff32', $FFFFDAB9 ));
16318:     clPeru32', $FFCD853F ));
16319:     clPlum32', $FFDDA0DD ));
16320:     clPowderBlue32', $FFB0E0E6 ));
16321:     clRosyBrown32', $FFBC8F8F ));
16322:     clRoyalBlue32', $FF4169E1 ));
16323:     clSaddleBrown32', $FF8B4513 ));
16324:     clSalmon32', $FFFA8072 ));
16325:     clSandyBrown32', $FFF4A460 ));
16326:     clSeaGreen32', $FF2E8B57 ));
16327:     clSeaShell32', $FFFFF5EE ));
16328:     clSienna32', $FFA0522D ));
16329:     clSilver32', $FFC0C0C0 ));
16330:     clSkyblue32', $FF87CEEB ));
16331:     clSlateBlue32', $FF6A5ACD ));
16332:     clSlateGray32', $FF708090 ));
16333:     clSlateGrey32', $FF708090 ));
16334:     clSnow32', $FFFFFAFA ));
16335:     clSpringgreen32', $FF00FF7F ));
16336:     clSteelblue32', $FF4682B4 ));
16337:     clTan32', $FFD2B48C ));
16338:     clThistle32', $FFD8BFD8 ));
16339:     clTomato32', $FFFF6347 ));
16340:     clTurquoise32', $FF40E0D0 ));
16341:     clViolet32', $FFEE82EE ));
16342:     clWheat32', $FFF5DEB3 ));
16343:     clWhitesmoke32', $FFF5F5F5 ));
16344:     clYellowgreen32', $FF9ACD32 ));
16345:     clTrWhite32', $7FFFFFFF ));
16346:     clTrBlack32', $7F000000 ));
16347:     clTrRed32', $7FFF0000 ));
16348:     clTrGreen32', $7F00FF00 ));
16349:     clTrBlue32', $7F0000FF ));
16350:   // Fixed point math constants
16351:   FixedOne = $10000;  FixedHalf = $7FFF;
16352:   FixedPI  = Round(PI * FixedOne);
16353:   FixedToFloat = 1/FixedOne;
16354:
16355:  Special Types
16356: ************************************************
16357:   type Complex = record
16358:     X, Y : Float;
16359:   end;
16360:   type TVector     = array of Float;
16361:   TIntVector  = array of Integer;
16362:   TCompVector = array of Complex;
16363:   TBoolVector = array of Boolean;
16364:   TStrVector  = array of String;
16365:   TMatrix     = array of TVector;
16366:   TIntMatrix  = array of TIntVector;
16367:   TCompMatrix = array of TCompVector;
16368:   TBoolMatrix = array of TBoolVector;
16369:   TStrMatrix  = array of TStrVector;
16370:   TByteArray = array[0..32767] of byte; !
16371:   THexArray = array [0..15] of Char; // = '0123456789ABCDEF';
16372:   TBitmapStyle = (bsNormal, bsCentered, bsStretched);
16373:   T2StringArray = array of array of string;
16374:   T2IntegerArray = array of array of integer;
16375:   AddTypeS('INT_PTR', 'Integer
16376:   AddTypeS('LONG_PTR', 'Integer
16377:   AddTypeS('UINT_PTR', 'Cardinal
16378:   AddTypeS('ULONG_PTR', 'Cardinal
16379:   AddTypeS('DWORD_PTR', 'ULONG_PTR
16380:   TIntegerDynArray', 'array of Integer
16381:   TCardinalDynArray', 'array of Cardinal
16382:   TWordDynArray', 'array of Word
16383:   TSmallIntDynArray', 'array of SmallInt
16384:   TByteDynArray', 'array of Byte
16385:   TShortIntDynArray', 'array of ShortInt
16386:   TInt64DynArray', 'array of Int64
16387:   TLongWordDynArray', 'array of LongWord
16388:   TSingleDynArray', 'array of Single
16389:   TDoubleDynArray', 'array of Double
16390:   TBooleanDynArray', 'array of Boolean
16391:   TStringDynArray', 'array of string
16392:   TWideStringDynArray', 'array of WideString
16393:   TDynByteArray     = array of Byte;
16394:   TDynShortintArray = array of Shortint;
16395:   TDynSmallintArray = array of Smallint;
16396:   TDynWordArray     = array of Word;
16397:   TDynIntegerArray  = array of Integer;
16398:   TDynLongintArray  = array of Longint;
16399:   TDynCardinalArray = array of Cardinal;
16400:   TDynInt64Array    = array of Int64;
16401:   TDynExtendedArray = array of Extended;
16402:   TDynDoubleArray   = array of Double;
16403:   TDynSingleArray   = array of Single;
16404:   TDynFloatArray    = array of Float;
```

```
16405:   TDynPointerArray  = array of Pointer;
16406:   TDynStringArray   = array of string;
16407:   TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
16408:     ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
16409:   TSynSearchOptions = set of TSynSearchOption;
16410:
16411:
16412:
16413: //* Project  : Base Include RunTime Lib for maXbox *Name: pas_includebox.inc
16414: ----------------------------------------------------------------------------
16415: procedure drawPolygon(vPoints: TXYVector; cFrm: TForm);
16416: procedure drawPlot(vPoints: TXYVector; cFrm: TForm; vcolor: integer);
16417: procedure SaveCanvas(vCanvas: TCanvas; FileName: string);
16418: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
16419: function CheckStringSum(vstring: string): integer;
16420: function HexToInt(HexNum: string): LongInt;
16421: function IntToBin(Int: Integer): String;
16422: function BinToInt(Binary: String): Integer;
16423: function HexToBin(HexNum: string): string; external2
16424: function BinToHex(Binary: String): string;
16425: function IntToFloat(i: Integer): double;
16426: function AddThousandSeparator(S: string; myChr: Char): string;
16427: function Max3(const X,Y,Z: Integer): Integer;
16428: procedure Swap(var X,Y: char); // faster without inline
16429: procedure ReverseString(var S: String);
16430: function CharToHexStr(Value: Char): string;
16431: function CharToUniCode(Value: Char): string;
16432: function Hex2Dec(Value: Str002): Byte;
16433: function HexStrCodeToStr(Value: string): string;
16434: function HexToStr(i: integer; value: string): string;
16435: function UniCodeToStr(Value: string): string;
16436: function CRC16(statement: string): string;
16437: function SearchForSubstrings(aStrList: TStrings; aSearchStr1, aSearchStr2: string): string;
16438: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string);
16439: procedure ShowInterfaces(myFile: string);
16440: function Fact2(av: integer): extended;
16441: Function BoolToStr(B: Boolean): string;
16442: Function GCD(x, y : LongInt) : LongInt;
16443: function LCM(m,n: longint): longint;
16444: function GetASCII: string;
16445: function GetItemHeight(Font: TFont): Integer;
16446: function myPlaySound(s: pchar; flag,syncflag: integer): boolean;
16447: function myGetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
16448: function getHINSTANCE: longword;
16449: function getHMODULE: longword;
16450: function GetASCII: string;
16451: function ByteIsOk(const AByte: string; var VB: Byte): boolean;
16452: function WordIsOk(const AWord: string; var VW: Word): boolean;
16453: function TwentyFourBitValueIsOk(const AValue: string; var VI: Integer): boolean;
16454: function LongIsOk(const ALong: string; var VC: Cardinal): boolean;
16455: function SafeStr(const s: string): string;
16456: function ExtractUrlPath(const FileName: string): string;
16457: function ExtractUrlName(const FileName: string): string;
16458: function IsInternet: boolean;
16459: function RotateLeft1Bit_u32( Value: uint32): uint32;
16460: procedure LinearRegression(const KnownY:array of Double;const KnownX: array of Double;NData:Int;var
       LF:TStLinEst; ErrorStats : Boolean);
16461: procedure getEnvironmentInfo;
16462: procedure AntiFreeze;
16463: function GetCPUSpeed: Double;
16464: function IsVirtualPcGuest : Boolean;
16465: function IsVmWareGuest : Boolean;
16466: procedure StartSerialDialog;
16467: function IsWoW64: boolean;
16468: function IsWow64String(var s: string): Boolean;
16469: procedure StartThreadDemo;
16470: Function RGB(R,G,B: Byte): TColor;
16471: Function Sendln(amess: string): boolean;
16472: Procedure maXbox;
16473: Function AspectRatio(aWidth, aHeight: Integer): String;
16474: function wget(aURL, afile: string): boolean;
16475: procedure PrintList(Value: TStringList);
16476: procedure PrintImage(aValue: TBitmap; Style: TBitmapStyle);
16477: procedure getEnvironmentInfo;
16478: procedure AntiFreeze;
16479: function getBitmap(apath: string): TBitmap;
16480: procedure ShowMessageBig(const aText : string);
16481: function YesNoDialog(const ACaption, AMsg: string): boolean;
16482: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);
16483: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
16484: //function myStrToBytes(const Value: String): TBytes;
16485: //function myBytesToStr(const Value: TBytes): String;
16486: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
16487: function getBitmap(apath: string): TBitmap;
16488: procedure ShowMessageBig(const aText : string);
16489: Function StrToBytes(const Value: String): TBytes;
16490: Function BytesToStr(const Value: TBytes): String;
16491: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
16492: function ReverseDNSLookup(const IPAdrs:String;const DNSServer:String;Timeout,Retries:Int;var
       HostName:String):Bool;
```

```
16493: function FindInPaths(const fileName, paths : String) : String;
16494: procedure initHexArray(var hexn: THexArray);
16495: function josephusG(n,k: integer; var graphout: string): integer;
16496: function isPowerof2(num: int64): boolean;
16497: function powerOf2(exponent: integer): int64;
16498: function getBigPI: string;
16499: procedure MakeSound(Frequency{Hz}, Duration{mSec}: Integer; Volume: TVolumeLevel; savefilePath: string);
16500:  function GetASCIILine: string;
16501: procedure MakeComplexSound(N:integer{stream # to use}; freqlist:TStrings; Duration{mSec}: Integer;
16502:                            pinknoise: boolean; shape: integer; Volume: TVolumeLevel);
16503: procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
16504: procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
16505: function mapfunc(ax, in_min, in_max, out_min, out_max: integer): integer;
16506: function mapmax(ax, in_min, in_max, out_min, out_max: integer): integer;
16507: function isKeypressed: boolean;
16508: function Keypress: boolean;
16509: procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStrings);
16510: function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
16511: function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
16512: function GetOSName: string;
16513: function GetOSVersion: string;
16514: function GetOSNumber: string;
16515: function getEnvironmentString: string;
16516: procedure StrReplace(var Str: String; Old, New: String);
16517: procedure SendEmail(mFrom,  mTo,  mSubject,  mBody,  mAttachment: variant);
16518: function getTeamViewerID: string;
16519: Procedure RecurseDirectory(Dir:String; IncludeSubs:boolean; callback:TFileCallbackProcedure);
16520: Procedure RecurseDirectory2(Dir : String; IncludeSubs : boolean);
16521: procedure WinInet_HttpGet(const Url: string; Stream:TStream);
16522: procedure GetQrCode2(Width,Height: Word;Correct_Level:string;const Data:string;apath:string);
16523: function StartSocketService: Boolean;
16524: procedure StartSocketServiceForm;
16525: function GetFileList(FileList: TStringlist; apath: string): TStringlist;
16526: function GetFileList1(apath: string): TStringlist;
16527: procedure LetFileList(FileList: TStringlist; apath: string);
16528: procedure StartWeb(aurl: string);
16529: function GetTodayFiles(startdir, amask: string): TStringlist;
16530: function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
16531: function JavahashCode(val: string): Integer;
16532: procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
16533: procedure SaveBytesToFile2(const Data: Sysutils.TBytes; const FileName: string);
16534: Procedure HideWindowForSeconds(secs: integer);   {//3 seconds}
16535: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm);   {//3 seconds}
16536: Procedure ConvertToGray(Cnv: TCanvas);
16537: function GetFileDate(aFile:string; aWithTime:Boolean):string;
16538: procedure ShowMemory;
16539: function ShowMemory2: string;
16540: function getHostIP: string;
16541: procedure ShowBitmap(bmap: TBitmap);
16542: function GetOsVersionInfo: TOSVersionInfo;      //thx to wischnewski
16543:
16544:
16545: // News of 3.9.8 up
16546: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
16547: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
16548: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
16549: JvChart - TJvChart Component - 2009 Public
16550: MemoryLeakReport in ini-file (MEMORYREPORT=Y)
16551: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
16552: TAdoQuery.SQL.Add() fixed, ShLwAPI extensions, Indy HTTPHeader Extensions
16553: DMath DLL included incl. Demos
16554: Interface Navigator menu/View/Intf Navigator
16555: Unit Explorer menu/Debug/Units Explorer
16556: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maXcel
16557: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
16558: Script History to 9 Files WebServer light /Options/Addons/WebServer
16559: Full Text Finder, JVSimLogic Simulator Package
16560: Halt-Stop Program in Menu, WebServer2, Stop Event ,
16561: Conversion Routines, Prebuild Forms, CodeSearch
16562: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
16563: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
16564: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
16565: JvChart - TJvChart Component - 2009 Public, mXGames, JvgXMLSerializer, TJvPaintFX
16566: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib
16567: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
16568: IDE Reflection API, Session Service Shell S3
16569: additional SynEdit API, isKeyPressed Routine, Bookmarks, OpenToolsAPI Catalog (OTAC)
16570: Class TMonitor, Configuration Tutorial maxbox_starter25.pdf, Chess.dll Game
16571: arduino map() function, PMRandom Generator
16572: StBarCode Lib, StreamReaderClass, BarCode Package, Astro Package
16573: more ShellAPI, add 32 more units, Simulated Annealing, GenAlgo
16574: REST Test Lib, Multilang Component, Forth Interpreter
16575: New Macros, Sendmail (instant email), DevCUnits, Tetris Addon
16576: DCOM, MDAC, MIDI, TLS support, Posmarks, Utils Addon
16577: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
16578: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
16579: First LCL of Lazarus, CmdLine API, ToDo List, 36 more Units preCompiled
16580: QRCode Service, add more CFunctions like CDateTime of Synapse
16581: Gamma Functions, IndyPackage4, HotLog Threadable, FormTemplateLibrary FTL
```

```
16582: Nonlinear regression, ADO Workbench Addon, Assign fixing, IntfNavigator fixing, Applet
16583: 30 more Units preCompiled, QRCode Indy Service, more CFunctions like CFill or SRand
16584: RestartDialog, RTF, SQL Scanner, RichEdit, 15 more Units
16585: Tool Section, SOAP Tester, Hot Log Logger2, TCPPortScan, 28 more Units
16586: BOLD Package, Indy Package5, maTRIx. MATHEMAX
16587: SPS Utils WDOS, Plc BitBus (PetriNet), 40 more units
16588: emax layers: system-package-component-unit-class-function-block
16589: HighPrecision Timers,  Indy Package6, AutoDetect, UltraForms
16590: Reversi, GOL, bugfixing, 8 more units, Tutorial 24 Clean Code
16591: Tutorial 18_3 RGB LED, OpenGL Geometry, maxpix, statictext
16592: OpenGL Game Demo: ..Options/Add Ons/Reversi
16593: IBUtils Refactor, InterBase Package, DotNet Routines (JvExControls)
16594: add 31 units, mX4 Introduction Paper, more Socket&Streams, ShortString Routines
16595: 7% performance gain (hot spot profiling)
16596: PEP -Pascal Education Program , GSM Module, CGI, PHP Runner
16597: add 42 + 22  (64 units), memcached database, autobookmark, Alcinoe PAC, IPC Lib
16598: Orpheus PAC, AsyncFree Library advapi32 samples, FirebirdExp+MySQL units
16599: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
16600:
16601: add routines in 3.9.7.5
16602: 097: procedure RIRegister_BarCodeScaner_Routines(S: TPSExec);
16603: 996: procedure RIRegister_DBCtrls_Routines(S: TPSExec);
16604: 069: procedure RIRegister_IdStrings_Routines(S: TPSExec);
16605: 516: procedure RIRegister_JclMultimedia_Routines(S: TPSExec);
16606: 215: procedure RIRegister_PNGLoader_Routines(S: TPSExec);
16607: 374: procedure RIRegister_SerDlgs_Routines(S: TPSExec);
16608: 777: procedure RIRegister_LinarBitmap_Routines(S: TPSExec);
16609:
16610: ////////////////////////// TestUnits //////////////////////////
16611:   SelftestPEM;
16612:   SelfTestCFundamentUtils;
16613:   SelfTestCFileUtils;
16614:   SelfTestCDateTime;
16615:   SelfTestCTimer;
16616:   SelfTestCRandom;
16617:   Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter
16618:             Assert(WinPathToUnixPath('\c\d.f') = '/c/d.f', 'WinPathToUnixPath
16619:
16620:   TGraphicControl = class(TControl)
16621:   private
16622:     FCanvas: TCanvas;
16623:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
16624:   protected
16625:     procedure Paint; virtual;
16626:     property Canvas: TCanvas read FCanvas;
16627:   public
16628:     constructor Create(AOwner: TComponent); override;
16629:     destructor Destroy; override;
16630:   end;
16631:
16632:   TCustomControl = class(TWinControl)
16633:   private
16634:     FCanvas: TCanvas;
16635:     procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
16636:   protected
16637:     procedure Paint; virtual;
16638:     procedure PaintWindow(DC: HDC); override;
16639:     property Canvas: TCanvas read FCanvas;
16640:   public
16641:     constructor Create(AOwner: TComponent); override;
16642:     destructor Destroy; override;
16643:   end;
16644:     RegisterPublishedProperties;
16645:     ('ONCHANGE', 'TNotifyEvent', iptrw);
16646:     ('ONCLICK', 'TNotifyEvent', iptrw);
16647:     ('ONDBLCLICK', 'TNotifyEvent', iptrw);
16648:     ('ONENTER', 'TNotifyEvent', iptrw);
16649:     ('ONEXIT', 'TNotifyEvent', iptrw);
16650:     ('ONKEYDOWN', 'TKeyEvent', iptrw);
16651:     ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
16652:     ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
16653:     ('ONMOUSEMOVE', 'TMouseMoveEvent', iptrw);
16654:     ('ONMOUSEUP', 'TMouseEvent', iptrw);
16655: //*************************************************************************
16656:   // To stop the while loop, click on Options/Show Include (boolean switch)!
16657:   Control a loop in a script with a form event:
16658:   IncludeON;   //control the while loop
16659:   while maxform1.ShowInclude1.checked do begin  //menu event Options/Show Include
16660:
16661: //-------------------------------------------------------------------------
16662: //*************mX4 ini-file Configuration************************************
16663: //-------------------------------------------------------------------------
16664:   using config file maxboxdef.ini       menu/Help/Config File
16665:
16666: //*** Definitions for maXbox mX3 ***
16667: [FORM]
16668: LAST_FILE=E:\maXbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
16669: FONTSIZE=14
16670: EXTENSION=txt
```

```
16671: SCREENX=1386
16672: SCREENY=1077
16673: MEMHEIGHT=350
16674: PRINTFONT=Courier New //GUI Settings
16675: LINENUMBERS=Y   //line numbers at gutter in editor at left side
16676: EXCEPTIONLOG=Y  //store excepts and success in 2 log files see below! – menu Debug/Show Last Exceptions
16677: EXECUTESHELL=Y  //prevents execution of ExecuteShell() or ExecuteCommand()
16678: BOOTSCRIPT=Y    //enabling load a boot script
16679: MEMORYREPORT=Y  //shows memory report on closing maXbox
16680: MACRO=Y         //expand macros (see below) in code e.g. #path:E:\maxbox\maxbox3\docs\
16681: NAVIGATOR=N     //shows function list at the right side of editor
16682: NAVWIDTH=350    //width of the right side interface list <CTRL L>
16683: AUTOBOOKMARK=Y  //sets on all functions a bookmark to jump
16684: [WEB]
16685: IPPORT=8080     //for internal webserver – menu /Options/Add Ons/WebServer
16686: IPHOST=192.168.1.53
16687: ROOTCERT=filepathY
16688: SCERT=filepathY
16689: RSAKEY=filepathY
16690: VERSIONCHECK=Y
16691:
16692: using Logfile: maxboxlog.log  , Exceptionlogfile: maxboxerrorlog.txt
16693:
16694: Also possible to set report memory in script to override ini setting
16695: procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
16696:
16697:
16698: After Change the ini file you can reload the file with ../Help/Config Update
16699:
16700: //-------------------------------------------------------------------------
16701: //**************mX4 maildef.ini ini-file Configuration*********************
16702: //-------------------------------------------------------------------------
16703: //*** Definitions for maXMail ***
16704: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
16705: [MAXMAIL]
16706: HOST=getmail.softwareschule.ch
16707: USER=mailusername
16708: PASS=password
16709: PORT=110
16710: SSL=Y
16711: BODY=Y
16712: LAST=5
16713:
16714: ADO Connection String:
16715: Provider=MSDASQL;DSN=mx3base;Uid=sa;Pwd=admin
16716:
16717:
16718: //-------------------------------------------------------------------------
16719: //**************mX4 Macro Tags  ********************************************
16720: //---------------------------------------------------------- ----------------
16721:
16722:   asm #name #hostmAPSN2APSN211le, #head,max: APSN21: 04.01.2014 19:05:50
      E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt end
16723:
16724: //Tag Macros
16725:
16726:   asm #name, #date, #host, #path, #file, #head, #sign #tech end
16727:
16728: //Tag Macros
16729: 10188: SearchAndCopy(memo1.lines, '#name', getUserNameWin, 11);
16730: 10189: SearchAndCopy(memo1.lines, '#date', datetimetoStr(now), 11);
16731: 10190: SearchAndCopy(memo1.lines, '#host', getComputernameWin, 11);
16732: 10191: SearchAndCopy(memo1.lines, '#path', fpath, 11);
16733: 10192: SearchAndCopy(memo1.lines, '#file', fname, 11);
16734: 10199  SearchAndCopy(memo1.lines, '#fils', fname +' '+SHA1(Act_Filename), 11);
16735: 10193: SearchAndCopy(memo1.lines, '#locs', intToStr(getCodeEnd), 11);
16736: 10194: SearchAndCopy(memo1.lines, '#perf', perftime, 11);
16737: 10195: SearchAndCopy(memo1.lines, '#sign', Format('%s: %s: %s',
16738:        [getUserNameWin, getComputernameWin, datetimetoStr(now),
16739: 10196: SearchAndCopy(memo1.lines, '#head',Format('%s: %s: %s %s ',
16740: 10197: [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
16741:        [getUserNameWin, getComputernameWin, datetimetoStr(now), Act_Filename]),11);
16742: 10198: SearchAndCopy(memo1.lines, '#tech',Format('perf: %s threads: %d %s %s',
16743:        [perftime, numprocessthreads, getIPAddress(getComputerNameWin), timetoStr(time), mbversion]), 11);
16744: //#tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
16745:
16746: //Replace Macros
16747:   SearchAndCopy(memo1.lines, '<TIME>', timetoStr(time), 6);
16748:   SearchAndCopy(memo1.lines, '<DATE>', datetoStr(date), 6);
16749:   SearchAndCopy(memo1.lines, '<PATH>', fpath, 6);
16750:   SearchAndCopy(memo1.lines, '<EXEPATH>', EXEPath, 9);
16751:   SearchAndCopy(memo1.lines, '<FILE>', fname, 6);
16752:   SearchAndCopy(memo1.lines, '<SOURCE>', ExePath+'Source', 8);
16753:
16754: 10198: SearchAndCopy(memo1.lines, '#tech'perf:  threads: 2 192.168.1.53 19:05:50 3.9.9.84
16755:        [perftime,numprocessthreads,getIPAddress(getComputerNameWin),timetoStr(time),mbversion]), 11);
16756: //#tech!84perf:  threads: 2 192.168.1.53 19:05:50 3.9.9.84
16757:   SearchAndCopy(memo1.lines, 'maxbox_extract_funclist399.txt
16758:
```

```
16759: //-------------------------------------------------------------------------------
16760: //**************mX4 ToDo List Tags  ../Help/ToDo List************************
16761: //------------------------------------------------------------ ---------------------
16762:
16763:     while I < sl.Count do begin
16764: //      if MatchesMask(sl[I], '*/? TODO ([a-z0-9_]*#[1-9]#)*:*') then
16765:       if MatchesMask(sl[I], '*/? TODO (?*#?#)*:*') then
16766:         BreakupToDo(Filename, sl, I, 'TODO', True, True) // full info TODO
16767:       else if MatchesMask(sl[I], '*/? DONE (?*#?#)*:*') then
16768:         BreakupToDo(Filename, sl, I, 'DONE', True, True) // full info DONE
16769:       else if MatchesMask(sl[I], '*/? TODO (#?#)*:*') then
16770:         BreakupToDo(Filename, sl, I, 'TODO', False, True) // only priority info TODO
16771:       else if MatchesMask(sl[I], '*/? DONE (#?#)*:*') then
16772:         BreakupToDo(Filename, sl, I, 'DONE', False, True) // only priority info DONE
16773:       else if MatchesMask(sl[I], '*/?*TODO*:*') then
16774:         BreakupToDo(Filename, sl, I, 'TODO', False, False) // custom TODO
16775:       else if MatchesMask(sl[I], '*/?*DONE*:*') then
16776:         BreakupToDo(Filename, sl, I, 'DONE', False, False); // custom DONE
16777:       Inc(I);
16778:     end;
16779:
16780:
16781: //-------------------------------------------------------------------------------
16782: //**************mX4 Public  Tools API ****************************************
16783: //-------------------------------------------------------------------------------
16784:   file : unit uPSI_fMain.pas;              {$OTAP} Open Tools API Catalog
16785:  // Those functions concern the editor and preprocessor, all of the IDE
16786:  Example: Call it with maxform1.Info1Click(self)
16787:  Note: Call all Methods with maxForm1., e.g.:
16788:                       maxForm1.ShellStyle1Click(self);
16789:
16790: procedure SIRegister_fMain(CL: TPSPascalCompiler);
16791: begin
16792:  Const('BYTECODE','String').SetString( 'bytecode.txt
16793:  Const('PSTEXT','String').SetString( 'PS Scriptfiles (*.txt)|*.TXT
16794:  Const('PSMODEL','String').SetString( 'PS Modelfiles (*.uc)|*.UC
16795:  Const('PSPASCAL','String').SetString( 'PS Pascalfiles (*.pas)|*.PAS
16796:  Const('PSINC','String').SetString( 'PS Includes (*.inc)|*.INC
16797:  Const('DEFFILENAME','String').SetString( 'firstdemo.txt
16798:  Const('DEFINIFILE','String').SetString( 'maxboxdef.ini
16799:  Const('EXCEPTLOGFILE','String').SetString( 'maxboxerrorlog.txt
16800:  Const('ALLFUNCTIONSLIST','String').SetString( 'upsi_allfunctionslist.txt
16801:  Const('ALLFUNCTIONSLISTPDF','String').SetString( 'maxbox_functions_all.pdf
16802:  Const('ALLOBJECTSLIST','String').SetString( 'docs\VCL.pdf
16803:  Const('ALLRESOURCELIST','String').SetString( 'docs\upsi_allresourcelist.txt
16804:  Const('INCLUDEBOX','String').SetString( 'pas_includebox.inc
16805:  Const('BOOTSCRIPT','String').SetString( 'maxbootscript.txt
16806:  Const('MBVERSION','String').SetString( '3.9.9.88
16807:  Const('VERSION','String').SetString('3.9.9.88
16808:  Const('MBVER','String').SetString( '399
16809:  Const('MBVERI','Integer').SetInt(399);
16810:  Const('MBVERIALL','Integer').SetInt(39988);
16811:  Const('EXENAME','String').SetString( 'maXbox3.exe
16812:  Const('MXSITE','String').SetString( 'http://www.softwareschule.ch/maxbox.htm
16813:  Const('MXVERSIONFILE','String').SetString( 'http://www.softwareschule.ch/maxvfile.txt
16814:  Const('MXINTERNETCHECK','String').SetString( 'www.ask.com
16815:  Const('MXMAIL','String').SetString( 'max@kleiner.com
16816:  Const('TAB','Char').SetString( #$09);
16817:  Const('CODECOMPLETION','String').SetString( 'bds_delphi.dci
16818:  SIRegister_TMaxForm1(CL);
16819: end;
16820:
16821:   with FindClass('TForm'),'TMaxForm1') do begin
16822:     memo2', 'TMemo', iptrw);
16823:     memo1', 'TSynMemo', iptrw);
16824:     CB1SCList', 'TComboBox', iptrw);
16825:     mxNavigator', 'TComboBox', iptrw);
16826:     IPHost', 'string', iptrw);
16827:     IPPort', 'integer', iptrw);
16828:     COMPort', 'integer', iptrw);      //3.9.6.4
16829:     Splitter1', 'TSplitter', iptrw);
16830:     PSScript', 'TPSScript', iptrw);
16831:     PS3DllPlugin', 'TPSDllPlugin', iptrw);
16832:     MainMenu1', 'TMainMenu', iptrw);
16833:     Program1', 'TMenuItem', iptrw);
16834:     Compile1', 'TMenuItem', iptrw);
16835:     Files1', 'TMenuItem', iptrw);
16836:     open1', 'TMenuItem', iptrw);
16837:     Save2', 'TMenuItem', iptrw);
16838:     Options1', 'TMenuItem', iptrw);
16839:     Savebefore1', 'TMenuItem', iptrw);
16840:     Largefont1', 'TMenuItem', iptrw);
16841:     sBytecode1', 'TMenuItem', iptrw);
16842:     Saveas3', 'TMenuItem', iptrw);
16843:     Clear1', 'TMenuItem', iptrw);
16844:     Slinenumbers1', 'TMenuItem', iptrw);
16845:     About1', 'TMenuItem', iptrw);
16846:     Search1', 'TMenuItem', iptrw);
16847:     SynPasSyn1', 'TSynPasSyn', iptrw);
```

```
16848:     memo1', 'TSynMemo', iptrw);
16849:     SynEditSearch1', 'TSynEditSearch', iptrw);
16850:     WordWrap1', 'TMenuItem', iptrw);
16851:     XPManifest1', 'TXPManifest', iptrw);
16852:     SearchNext1', 'TMenuItem', iptrw);
16853:     Replace1', 'TMenuItem', iptrw);
16854:     PSImport_Controls1', 'TPSImport_Controls', iptrw);
16855:     PSImport_Classes1', 'TPSImport_Classes', iptrw);
16856:     ShowInclude1', 'TMenuItem', iptrw);
16857:     SynEditPrint1', 'TSynEditPrint', iptrw);
16858:     Printout1', 'TMenuItem', iptrw);
16859:     mnPrintColors1', 'TMenuItem', iptrw);
16860:     dlgFilePrint', 'TPrintDialog', iptrw);
16861:     dlgPrintFont1', 'TFontDialog', iptrw);
16862:     mnuPrintFont1', 'TMenuItem', iptrw);
16863:     Include1', 'TMenuItem', iptrw);
16864:     CodeCompletionList1', 'TMenuItem', iptrw);
16865:     IncludeList1', 'TMenuItem', iptrw);
16866:     ImageList1', 'TImageList', iptrw);
16867:     ImageList2', 'TImageList', iptrw);
16868:     CoolBar1', 'TCoolBar', iptrw);
16869:     ToolBar1', 'TToolBar', iptrw);
16870:     tbtnLoad', 'TToolButton', iptrw);
16871:     ToolButton2', 'TToolButton', iptrw);
16872:     tbtnFind', 'TToolButton', iptrw);
16873:     tbtnCompile', 'TToolButton', iptrw);
16874:     tbtnTrans', 'TToolButton', iptrw);
16875:     tbtnUseCase', 'TToolButton', iptrw);   //3.8
16876:     toolbtnTutorial', 'TToolButton', iptrw);
16877:     tbtn6res', 'TToolButton', iptrw);
16878:     ToolButton5', 'TToolButton', iptrw);
16879:     ToolButton1', 'TToolButton', iptrw);
16880:     ToolButton3', 'TToolButton', iptrw);
16881:     statusBar1', 'TStatusBar', iptrw);
16882:     SaveOutput1', 'TMenuItem', iptrw);
16883:     ExportClipboard1', 'TMenuItem', iptrw);
16884:     Close1', 'TMenuItem', iptrw);
16885:     Manual1', 'TMenuItem', iptrw);
16886:     About2', 'TMenuItem', iptrw);
16887:     loadLastfile1', 'TMenuItem', iptrw);
16888:     imglogo', 'TImage', iptrw);
16889:     cedebug', 'TPSScriptDebugger', iptrw);
16890:     debugPopupMenu1', 'TPopupMenu', iptrw);
16891:     BreakPointMenu', 'TMenuItem', iptrw);
16892:     Decompile1', 'TMenuItem', iptrw);
16893:     StepInto1', 'TMenuItem', iptrw);
16894:     StepOut1', 'TMenuItem', iptrw);
16895:     Reset1', 'TMenuItem', iptrw);
16896:     DebugRun1', 'TMenuItem', iptrw);
16897:     PSImport_ComObj1', 'TPSImport_ComObj', iptrw);
16898:     PSImport_StdCtrls1', 'TPSImport_StdCtrls', iptrw);
16899:     PSImport_Forms1', 'TPSImport_Forms', iptrw);
16900:     PSImport_DateUtils1', 'TPSImport_DateUtils', iptrw);
16901:     tutorial4', 'TMenuItem', iptrw);
16902:     ExporttoClipboard1', 'TMenuItem', iptrw);
16903:     ImportfromClipboard1', 'TMenuItem', iptrw);
16904:     N4', 'TMenuItem', iptrw);
16905:     N5', 'TMenuItem', iptrw);
16906:     N6', 'TMenuItem', iptrw);
16907:     ImportfromClipboard2', 'TMenuItem', iptrw);
16908:     tutorial1', 'TMenuItem', iptrw);
16909:     N7', 'TMenuItem', iptrw);
16910:     ShowSpecChars1', 'TMenuItem', iptrw);
16911:     OpenDirectory1', 'TMenuItem', iptrw);
16912:     procMess', 'TMenuItem', iptrw);
16913:     tbtnUseCase', 'TToolButton', iptrw);
16914:     ToolButton7', 'TToolButton', iptrw);
16915:     EditFont1', 'TMenuItem', iptrw);
16916:     UseCase1', 'TMenuItem', iptrw);
16917:     tutorial21', 'TMenuItem', iptrw);
16918:     OpenUseCase1', 'TMenuItem', iptrw);
16919:     PSImport_DB1', 'TPSImport_DB', iptrw);
16920:     tutorial31', 'TMenuItem', iptrw);
16921:     SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
16922:     HTMLSyntax1', 'TMenuItem', iptrw);
16923:     ShowInterfaces1', 'TMenuItem', iptrw);
16924:     Tutorial5', 'TMenuItem', iptrw);
16925:     AllFunctionsList1', 'TMenuItem', iptrw);
16926:     ShowLastException1', 'TMenuItem', iptrw);
16927:     PlayMP31', 'TMenuItem', iptrw);
16928:     SynTeXSyn1', 'TSynTeXSyn', iptrw);
16929:     texSyntax1', 'TMenuItem', iptrw);
16930:     N8', 'TMenuItem', iptrw);
16931:     GetEMails1', 'TMenuItem', iptrw);
16932:     SynCppSyn1', 'TSynCppSyn', iptrw);
16933:     CSyntax1', 'TMenuItem', iptrw);
16934:     Tutorial6', 'TMenuItem', iptrw);
16935:     New1', 'TMenuItem', iptrw);
16936:     AllObjectsList1', 'TMenuItem', iptrw);
```

```
16937:     LoadBytecode1', 'TMenuItem', iptrw);
16938:     CipherFile1', 'TMenuItem', iptrw);
16939:     N9', 'TMenuItem', iptrw);
16940:     N10', 'TMenuItem', iptrw);
16941:     Tutorial11', 'TMenuItem', iptrw);
16942:     Tutorial71', 'TMenuItem', iptrw);
16943:     UpdateService1', 'TMenuItem', iptrw);
16944:     PascalSchool1', 'TMenuItem', iptrw);
16945:     Tutorial81', 'TMenuItem', iptrw);
16946:     DelphiSite1', 'TMenuItem', iptrw);
16947:     Output1', 'TMenuItem', iptrw);
16948:     TerminalStyle1', 'TMenuItem', iptrw);
16949:     ReadOnly1', 'TMenuItem', iptrw);
16950:     ShellStyle1', 'TMenuItem', iptrw);
16951:     BigScreen1', 'TMenuItem', iptrw);
16952:     Tutorial91', 'TMenuItem', iptrw);
16953:     SaveOutput2', 'TMenuItem', iptrw);
16954:     N11', 'TMenuItem', iptrw);
16955:     SaveScreenshot', 'TMenuItem', iptrw);
16956:     Tutorial101', 'TMenuItem', iptrw);
16957:     SQLSyntax1', 'TMenuItem', iptrw);
16958:     SynSQLSyn1', 'TSynSQLSyn', iptrw);
16959:     Console1', 'TMenuItem', iptrw);
16960:     SynXMLSyn1', 'TSynXMLSyn', iptrw);
16961:     XMLSyntax1', 'TMenuItem', iptrw);
16962:     ComponentCount1', 'TMenuItem', iptrw);
16963:     NewInstance1', 'TMenuItem', iptrw);
16964:     toolbtnTutorial', 'TToolButton', iptrw);
16965:     Memory1', 'TMenuItem', iptrw);
16966:     SynJavaSyn1', 'TSynJavaSyn', iptrw);
16967:     JavaSyntax1', 'TMenuItem', iptrw);
16968:     SyntaxCheck1', 'TMenuItem', iptrw);
16969:     Tutorial10Statistics1', 'TMenuItem', iptrw);
16970:     ScriptExplorer1', 'TMenuItem', iptrw);
16971:     FormOutput1', 'TMenuItem', iptrw);
16972:     ArduinoDump1', 'TMenuItem', iptrw);
16973:     AndroidDump1', 'TMenuItem', iptrw);
16974:     GotoEnd1', 'TMenuItem', iptrw);
16975:     AllResourceList1', 'TMenuItem', iptrw);
16976:     ToolButton4', 'TToolButton', iptrw);
16977:     tbtn6res', 'TToolButton', iptrw);
16978:     Tutorial11Forms1', 'TMenuItem', iptrw);
16979:     Tutorial12SQL1', 'TMenuItem', iptrw);
16980:     ResourceExplore1', 'TMenuItem', iptrw);
16981:     Info1', 'TMenuItem', iptrw);
16982:     N12', 'TMenuItem', iptrw);
16983:     CryptoBox1', 'TMenuItem', iptrw);
16984:     Tutorial13Ciphering1', 'TMenuItem', iptrw);
16985:     CipherFile2', 'TMenuItem', iptrw);
16986:     N13', 'TMenuItem', iptrw);
16987:     ModulesCount1', 'TMenuItem', iptrw);
16988:     AddOns2', 'TMenuItem', iptrw);
16989:     N4GewinntGame1', 'TMenuItem', iptrw);
16990:     DocuforAddOns1', 'TMenuItem', iptrw);
16991:     Tutorial14Async1', 'TMenuItem', iptrw);
16992:     Lessons15Review1', 'TMenuItem', iptrw);
16993:     SynPHPSyn1', 'TSynPHPSyn', iptrw);
16994:     PHPSyntax1', 'TMenuItem', iptrw);
16995:     Breakpoint1', 'TMenuItem', iptrw);
16996:     SerialRS2321', 'TMenuItem', iptrw);
16997:     N14', 'TMenuItem', iptrw);
16998:     SynCSSyn1', 'TSynCSSyn', iptrw);
16999:     CSyntax2', 'TMenuItem', iptrw);
17000:     Calculator1', 'TMenuItem', iptrw);
17001:     tbtnSerial', 'TToolButton', iptrw);
17002:     ToolButton8', 'TToolButton', iptrw);
17003:     Tutorial151', 'TMenuItem', iptrw);
17004:     N15', 'TMenuItem', iptrw);
17005:     N16', 'TMenuItem', iptrw);
17006:     ControlBar1', 'TControlBar', iptrw);
17007:     ToolBar2', 'TToolBar', iptrw);
17008:     BtnOpen', 'TToolButton', iptrw);
17009:     BtnSave', 'TToolButton', iptrw);
17010:     BtnPrint', 'TToolButton', iptrw);
17011:     BtnColors', 'TToolButton', iptrw);
17012:     btnClassReport', 'TToolButton', iptrw);
17013:     BtnRotateRight', 'TToolButton', iptrw);
17014:     BtnFullSize', 'TToolButton', iptrw);
17015:     BtnFitToWindowSize', 'TToolButton', iptrw);
17016:     BtnZoomMinus', 'TToolButton', iptrw);
17017:     BtnZoomPlus', 'TToolButton', iptrw);
17018:     Panel1', 'TPanel', iptrw);
17019:     LabelBrettgroesse', 'TLabel', iptrw);
17020:     CB1SCList', 'TComboBox', iptrw);
17021:     ImageListNormal', 'TImageList', iptrw);
17022:     spbtnexplore', 'TSpeedButton', iptrw);
17023:     spbtnexample', 'TSpeedButton', iptrw);
17024:     spbsaveas', 'TSpeedButton', iptrw);
17025:     imglogobox', 'TImage', iptrw);
```

```
17026:     EnlargeFont1', 'TMenuItem', iptrw);
17027:     EnlargeFont2', 'TMenuItem', iptrw);
17028:     ShrinkFont1', 'TMenuItem', iptrw);
17029:     ThreadDemo1', 'TMenuItem', iptrw);
17030:     HEXEditor1', 'TMenuItem', iptrw);
17031:     HEXView1', 'TMenuItem', iptrw);
17032:     HEXInspect1', 'TMenuItem', iptrw);
17033:     SynExporterHTML1', 'TSynExporterHTML', iptrw);
17034:     ExporttoHTML1', 'TMenuItem', iptrw);
17035:     ClassCount1', 'TMenuItem', iptrw);
17036:     HTMLOutput1', 'TMenuItem', iptrw);
17037:     HEXEditor2', 'TMenuItem', iptrw);
17038:     Minesweeper1', 'TMenuItem', iptrw);
17039:     N17', 'TMenuItem', iptrw);
17040:     PicturePuzzle1', 'TMenuItem', iptrw);
17041:     sbvclhelp', 'TSpeedButton', iptrw);
17042:     DependencyWalker1', 'TMenuItem', iptrw);
17043:     WebScanner1', 'TMenuItem', iptrw);
17044:     View1', 'TMenuItem', iptrw);
17045:     mnToolbar1', 'TMenuItem', iptrw);
17046:     mnStatusbar2', 'TMenuItem', iptrw);
17047:     mnConsole2', 'TMenuItem', iptrw);
17048:     mnCoolbar2', 'TMenuItem', iptrw);
17049:     mnSplitter2', 'TMenuItem', iptrw);
17050:     WebServer1', 'TMenuItem', iptrw);
17051:     Tutorial17Server1', 'TMenuItem', iptrw);
17052:     Tutorial18Arduino1', 'TMenuItem', iptrw);
17053:     SynPerlSyn1', 'TSynPerlSyn', iptrw);
17054:     PerlSyntax1', 'TMenuItem', iptrw);
17055:     SynPythonSyn1', 'TSynPythonSyn', iptrw);
17056:     PythonSyntax1', 'TMenuItem', iptrw);
17057:     DMathLibrary1', 'TMenuItem', iptrw);
17058:     IntfNavigator1', 'TMenuItem', iptrw);
17059:     EnlargeFontConsole1', 'TMenuItem', iptrw);
17060:     ShrinkFontConsole1', 'TMenuItem', iptrw);
17061:     SetInterfaceList1', 'TMenuItem', iptrw);
17062:     popintfList', 'TPopupMenu', iptrw);
17063:     intfAdd1', 'TMenuItem', iptrw);
17064:     intfDelete1', 'TMenuItem', iptrw);
17065:     intfRefactor1', 'TMenuItem', iptrw);
17066:     Defactor1', 'TMenuItem', iptrw);
17067:     Tutorial19COMArduino1', 'TMenuItem', iptrw);
17068:     Tutorial20Regex', 'TMenuItem', iptrw);
17069:     N18', 'TMenuItem', iptrw);
17070:     ManualE1', 'TMenuItem', iptrw);
17071:     FullTextFinder1', 'TMenuItem', iptrw);
17072:     Move1', 'TMenuItem', iptrw);
17073:     FractalDemo1', 'TMenuItem', iptrw);
17074:     Tutorial21Android1', 'TMenuItem', iptrw);
17075:     Tutorial0Function1', 'TMenuItem', iptrw);
17076:     SimuLogBox1', 'TMenuItem', iptrw);
17077:     OpenExamples1', 'TMenuItem', iptrw);
17078:     SynJScriptSyn1', 'TSynJScriptSyn', iptrw);
17079:     JavaScriptSyntax1', 'TMenuItem', iptrw);
17080:     Halt1', 'TMenuItem', iptrw);
17081:     CodeSearch1', 'TMenuItem', iptrw);
17082:     SynRubySyn1', 'TSynRubySyn', iptrw);
17083:     RubySyntax1', 'TMenuItem', iptrw);
17084:     Undo1', 'TMenuItem', iptrw);
17085:     SynUNIXShellScriptSyn1', 'TSynUNIXShellScriptSyn', iptrw);
17086:     LinuxShellScript1', 'TMenuItem', iptrw);
17087:     Rename1', 'TMenuItem', iptrw);
17088:     spdcodesearch', 'TSpeedButton', iptrw);
17089:     Preview1', 'TMenuItem', iptrw);
17090:     Tutorial22Services1', 'TMenuItem', iptrw);
17091:     Tutorial23RealTime1', 'TMenuItem', iptrw);
17092:     Configuration1', 'TMenuItem', iptrw);
17093:     MP3Player1', 'TMenuItem', iptrw);
17094:     DLLSpy1', 'TMenuItem', iptrw);
17095:     SynURIOpener1', 'TSynURIOpener', iptrw);
17096:     SynURISyn1', 'TSynURISyn', iptrw);
17097:     URILinksClicks1', 'TMenuItem', iptrw);
17098:     EditReplace1', 'TMenuItem', iptrw);
17099:     GotoLine1', 'TMenuItem', iptrw);
17100:     ActiveLineColor1', 'TMenuItem', iptrw);
17101:     ConfigFile1', 'TMenuItem', iptrw);
17102:     Sort1Intflist', 'TMenuItem', iptrw);
17103:     Redo1', 'TMenuItem', iptrw);
17104:     Tutorial24CleanCode1', 'TMenuItem', iptrw);
17105:     Tutorial25Configuration1', 'TMenuItem', iptrw);
17106:     IndentSelection1', 'TMenuItem', iptrw);
17107:     UnindentSection1', 'TMenuItem', iptrw);
17108:     SkyStyle1', 'TMenuItem', iptrw);
17109:     N19', 'TMenuItem', iptrw);
17110:     CountWords1', 'TMenuItem', iptrw);
17111:     imbookmarkimages', 'TImageList', iptrw);
17112:     Bookmark11', 'TMenuItem', iptrw);
17113:     N20', 'TMenuItem', iptrw);
17114:     Bookmark21', 'TMenuItem', iptrw);
```

```
17115:     Bookmark31', 'TMenuItem', iptrw);
17116:     Bookmark41', 'TMenuItem', iptrw);
17117:     SynMultiSyn1', 'TSynMultiSyn', iptrw);
17118:
17119:     Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
17120:     Procedure IFPS3ClassesPlugin1ExecImport(Sender: TObject; Exec: TPSExec; x:TPSRuntimeClassImporter);
17121:     Procedure PSScriptCompile( Sender : TPSScript)
17122:     Procedure Compile1Click( Sender : TObject)
17123:     Procedure PSScriptExecute( Sender : TPSScript)
17124:     Procedure open1Click( Sender : TObject)
17125:     Procedure Save2Click( Sender : TObject)
17126:     Procedure Savebefore1Click( Sender : TObject)
17127:     Procedure Largefont1Click( Sender : TObject)
17128:     Procedure FormActivate( Sender : TObject)
17129:     Procedure SBytecode1Click( Sender : TObject)
17130:     Procedure FormKeyPress( Sender : TObject; var Key : Char)
17131:     Procedure Saveas3Click( Sender : TObject)
17132:     Procedure Clear1Click( Sender : TObject)
17133:     Procedure Slinenumbers1Click( Sender : TObject)
17134:     Procedure About1Click( Sender : TObject)
17135:     Procedure Search1Click( Sender : TObject)
17136:     Procedure FormCreate( Sender : TObject)
17137:     Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Integer;
17138:                                                 var Action : TSynReplaceAction)
17139:     Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
17140:     Procedure WordWrap1Click( Sender : TObject)
17141:     Procedure SearchNext1Click( Sender : TObject)
17142:     Procedure Replace1Click( Sender : TObject)
17143:     Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FName,Output:String):Bool;
17144:     Procedure ShowInclude1Click( Sender : TObject)
17145:     Procedure Printout1Click( Sender : TObject)
17146:     Procedure mnuPrintFont1Click( Sender : TObject)
17147:     Procedure Include1Click( Sender : TObject)
17148:     Procedure FormDestroy( Sender : TObject)
17149:     Procedure FormClose( Sender : TObject; var Action : TCloseAction)
17150:     Procedure UpdateView1Click( Sender : TObject)
17151:     Procedure CodeCompletionList1Click( Sender : TObject)
17152:     Procedure SaveOutput1Click( Sender : TObject)
17153:     Procedure ExportClipboard1Click( Sender : TObject)
17154:     Procedure Close1Click( Sender : TObject)
17155:     Procedure Manual1Click( Sender : TObject)
17156:     Procedure LoadLastFile1Click( Sender : TObject)
17157:     Procedure Memo1Change( Sender : TObject)
17158:     Procedure Decompile1Click( Sender : TObject)
17159:     Procedure StepInto1Click( Sender : TObject)
17160:     Procedure StepOut1Click( Sender : TObject)
17161:     Procedure Reset1Click( Sender : TObject)
17162:     Procedure cedebugAfterExecute( Sender : TPSScript)
17163:     Procedure cedebugBreakpoint(Sender:TObject; const FileName:String; Position,Row, Col: Cardinal)
17164:     Procedure cedebugCompile( Sender : TPSScript)
17165:     Procedure cedebugExecute( Sender : TPSScript)
17166:     Procedure cedebugIdle( Sender : TObject)
17167:     Procedure cedebugLineInfo( Sender:TObject;const FileName:String; Position, Row, Col : Cardinal)
17168:     Procedure Memo1SpecialLineColors(Sender: TObject; Line:Int; var Special:Boolean;var FG,BG:TColor);
17169:     Procedure BreakPointMenuClick( Sender : TObject)
17170:     Procedure DebugRun1Click( Sender : TObject)
17171:     Procedure tutorial4Click( Sender : TObject)
17172:     Procedure ImportfromClipboard1Click( Sender : TObject)
17173:     Procedure ImportfromClipboard2Click( Sender : TObject)
17174:     Procedure tutorial1Click( Sender : TObject)
17175:     Procedure ShowSpecChars1Click( Sender : TObject)
17176:     Procedure StatusBar1DblClick( Sender : TObject)
17177:     Procedure PSScriptLine( Sender : TObject)
17178:     Procedure OpenDirectory1Click( Sender : TObject)
17179:     Procedure procMessClick( Sender : TObject)
17180:     Procedure tbtnUseCaseClick( Sender : TObject)
17181:     Procedure EditFont1Click( Sender : TObject)
17182:     Procedure tutorial21Click( Sender : TObject)
17183:     Procedure tutorial31Click( Sender : TObject)
17184:     Procedure HTMLSyntax1Click( Sender : TObject)
17185:     Procedure ShowInterfaces1Click( Sender : TObject)
17186:     Procedure Tutorial5Click( Sender : TObject)
17187:     Procedure ShowLastException1Click( Sender : TObject)
17188:     Procedure PlayMP31Click( Sender : TObject)
17189:     Procedure AllFunctionsList1Click( Sender : TObject)
17190:     Procedure texSyntax1Click( Sender : TObject)
17191:     Procedure GetEMails1Click( Sender : TObject)
17192:     procedure DelphiSite1Click(Sender: TObject);
17193:     procedure TerminalStyle1Click(Sender: TObject);
17194:     procedure ReadOnly1Click(Sender: TObject);
17195:     procedure ShellStyle1Click(Sender: TObject);
17196:     procedure Console1Click(Sender: TObject);     //3.2
17197:     procedure BigScreen1Click(Sender: TObject);
17198:     procedure Tutorial91Click(Sender: TObject);
17199:     procedure SaveScreenshotClick(Sender: TObject);
17200:     procedure Tutorial101Click(Sender: TObject);
17201:     procedure SQLSyntax1Click(Sender: TObject);
17202:     procedure XMLSyntax1Click(Sender: TObject);
17203:     procedure ComponentCount1Click(Sender: TObject);
```

```
17204:     procedure NewInstance1Click(Sender: TObject);
17205:     procedure CSyntax1Click(Sender: TObject);
17206:     procedure Tutorial6Click(Sender: TObject);
17207:     procedure New1Click(Sender: TObject);
17208:     procedure AllObjectsList1Click(Sender: TObject);
17209:     procedure LoadBytecode1Click(Sender: TObject);
17210:     procedure CipherFile1Click(Sender: TObject);  //V3.5
17211:     procedure NewInstance1Click(Sender: TObject);
17212:     procedure toolbtnTutorialClick(Sender: TObject);
17213:     procedure Memory1Click(Sender: TObject);
17214:     procedure JavaSyntax1Click(Sender: TObject);
17215:     procedure SyntaxCheck1Click(Sender: TObject);
17216:     procedure ScriptExplorer1Click(Sender: TObject);
17217:     procedure FormOutput1Click(Sender: TObject);  //V3.6
17218:     procedure GotoEnd1Click(Sender: TObject);
17219:     procedure AllResourceList1Click(Sender: TObject);
17220:     procedure tbtn6resClick(Sender: TObject);   //V3.7
17221:     procedure Info1Click(Sender: TObject);
17222:     procedure Tutorial10Statistics1Click(Sender: TObject);
17223:     procedure Tutorial11Forms1Click(Sender: TObject);
17224:     procedure Tutorial12SQL1Click(Sender: TObject);  //V3.8
17225:     procedure ResourceExplore1Click(Sender: TObject);
17226:     procedure Info1Click(Sender: TObject);
17227:     procedure CryptoBox1Click(Sender: TObject);
17228:     procedure ModulesCount1Click(Sender: TObject);
17229:     procedure N4GewinntGame1Click(Sender: TObject);
17230:     procedure PHPSyntax1Click(Sender: TObject);
17231:     procedure SerialRS2321Click(Sender: TObject);
17232:     procedure CSyntax2Click(Sender: TObject);
17233:     procedure Calculator1Click(Sender: TObject);
17234:     procedure Tutorial13Ciphering1Click(Sender: TObject);
17235:     procedure Tutorial14Async1Click(Sender: TObject);
17236:     procedure PHPSyntax1Click(Sender: TObject);
17237:     procedure BtnZoomPlusClick(Sender: TObject);
17238:     procedure BtnZoomMinusClick(Sender: TObject);
17239:     procedure btnClassReportClick(Sender: TObject);
17240:     procedure ThreadDemo1Click(Sender: TObject);
17241:     procedure HEXView1Click(Sender: TObject);
17242:     procedure ExporttoHTML1Click(Sender: TObject);
17243:     procedure Minesweeper1Click(Sender: TObject);
17244:     procedure PicturePuzzle1Click(Sender: TObject); //V3.9
17245:     procedure sbvclhelpClick(Sender: TObject);
17246:     procedure DependencyWalker1Click(Sender: TObject);
17247:     procedure CB1SCListDrawItem(Control:TWinControl;Index:Int;aRect:TRect;State:TOwnerDrawState);
17248:     procedure WebScanner1Click(Sender: TObject);
17249:     procedure mnToolbar1Click(Sender: TObject);
17250:     procedure mnStatusbar2Click(Sender: TObject);
17251:     procedure mnConsole2Click(Sender: TObject);
17252:     procedure mnCoolbar2Click(Sender: TObject);
17253:     procedure mnSplitter2Click(Sender: TObject);
17254:     procedure WebServer1Click(Sender: TObject);
17255:     procedure PerlSyntax1Click(Sender: TObject);
17256:     procedure PythonSyntax1Click(Sender: TObject);
17257:     procedure DMathLibrary1Click(Sender: TObject);
17258:     procedure IntfNavigator1Click(Sender: TObject);
17259:     procedure FullTextFinder1Click(Sender: TObject);
17260:     function AppName: string;
17261:     function ScriptName: string;
17262:     function LastName: string;
17263:     procedure FractalDemo1Click(Sender: TObject);
17264:     procedure SimuLogBox1Click(Sender: TObject);
17265:     procedure OpenExamples1Click(Sender: TObject);
17266:     procedure Halt1Click(Sender: TObject);
17267:     procedure Stop;
17268:     procedure CodeSearch1Click(Sender: TObject);
17269:     procedure RubySyntax1Click(Sender: TObject);
17270:     procedure Undo1Click(Sender: TObject);
17271:     procedure LinuxShellScript1Click(Sender: TObject);
17272:     procedure WebScannerDirect(urls: string);
17273:     procedure WebScanner(urls: string);
17274:     procedure LoadInterfaceList2;
17275:     procedure DLLSpy1Click(Sender: TObject);
17276:     procedure Memo1DblClick(Sender: TObject);
17277:     procedure URILinksClicks1Click(Sender: TObject);
17278:     procedure GotoLine1Click(Sender: TObject);
17279:     procedure ConfigFile1Click(Sender: TObject);
17280:     Procedure Sort1IntflistClick( Sender : TObject)
17281:     Procedure Redo1Click( Sender : TObject)
17282:     Procedure Tutorial24CleanCode1Click( Sender : TObject)
17283:     Procedure IndentSelection1Click( Sender : TObject)
17284:     Procedure UnindentSection1Click( Sender : TObject)
17285:     Procedure SkyStyle1Click( Sender : TObject)
17286:     Procedure CountWords1Click( Sender : TObject)
17287:     Procedure Memo1PlaceBookmark( Sender : TObject; var Mark : TSynEditMark)
17288:     Procedure Memo1GutterClick(Sender:TObject;Button:TMouseButton;X,Y,Line:Integer;Mark:TSynEditMark);
17289:     Procedure Bookmark11Click( Sender : TObject)
17290:     Procedure Bookmark21Click( Sender : TObject)
17291:     Procedure Bookmark31Click( Sender : TObject)
17292:     Procedure Bookmark41Click( Sender : TObject)
```

```
17293:     Procedure SynMultiSyn1CustomRange(Sender:TSynMultiSyn;Operation:TRangeOperation;var Range:Pointer);
17294:     'STATMemoryReport', 'boolean', iptrw);
17295:     'IPPort', 'integer', iptrw);
17296:     'COMPort', 'integer', iptrw);
17297:     'lbintflist', 'TListBox', iptrw);
17298:     Function GetStatChange : boolean
17299:     Procedure SetStatChange( vstat : boolean)
17300:     Function GetActFileName : string
17301:     Procedure SetActFileName( vname : string)
17302:     Function GetLastFileName : string
17303:     Procedure SetLastFileName( vname : string)
17304:     Procedure WebScannerDirect( urls : string)
17305:     Procedure LoadInterfaceList2
17306:     Function GetStatExecuteShell : boolean
17307:     Procedure DoEditorExecuteCommand( EditorCommand : word)
17308:     function GetActiveLineColor: TColor
17309:     procedure SetActiveLineColor(acolor: TColor)
17310:     procedure ScriptListbox1Click(Sender: TObject);
17311:     procedure Memo2KeyPress(Sender: TObject; var Key: Char);
17312:     procedure EnlargeGutter1Click(Sender: TObject);
17313:     procedure Tetris1Click(Sender: TObject);
17314:     procedure ToDoList1Click(Sender: TObject);
17315:     procedure ProcessList1Click(Sender: TObject);
17316:     procedure MetricReport1Click(Sender: TObject);
17317:     procedure ProcessList1Click(Sender: TObject);
17318:     procedure TCPSockets1Click(Sender: TObject);
17319:     procedure ConfigUpdate1Click(Sender: TObject);
17320:     procedure ADOWorkbench1Click(Sender: TObject);
17321:     procedure SocketServer1Click(Sender: TObject);
17322:     procedure FormDemo1Click(Sender: TObject);
17323:     procedure Richedit1Click(Sender: TObject);
17324:     procedure SimpleBrowser1Click(Sender: TObject);
17325:     procedure DOSShell1Click(Sender: TObject);
17326:     procedure SynExport1Click(Sender: TObject);
17327:     procedure ExporttoRTF1Click(Sender: TObject);
17328:     procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
17329:     procedure SOAPTester1Click(Sender: TObject);
17330:     procedure Sniffer1Click(Sender: TObject);
17331:     procedure AutoDetectSyntax1Click(Sender: TObject);
17332:     procedure FPlot1Click(Sender: TObject);
17333:     procedure PasStyle1Click(Sender: TObject);
17334:     procedure Tutorial183RGBLED1Click(Sender: TObject);
17335:     procedure Reversi1Click(Sender: TObject);
17336:     procedure ManualmaXbox1Click(Sender: TObject);
17337:     procedure BlaisePascalMagazine1Click(Sender: TObject);
17338:     procedure AddToDo1Click(Sender: TObject);
17339:     procedure CreateGUID1Click(Sender: TObject);
17340:     procedure Tutorial27XML1Click(Sender: TObject);
17341:     procedure CreateDLLStub1Click(Sender: TObject);
17342:
17343:
17344: //----------------------------------------------------------------------------
17345: //*************mX4 Editor SynEdit Tools API ********************************
17346: //----------------------------------------------------------------------------
17347: (*----------------------------------------------------------------------------*)
17348: procedure SIRegister_TCustomSynEdit(CL: TPSPascalCompiler);
17349: begin
17350:   //with RegClassS(CL,'TCustomControl', 'TCustomSynEdit') do
17351:   with FindClass('TCustomControl'),'TCustomSynEdit') do begin
17352:     Constructor Create( AOwner : TComponent)
17353:     SelStart', 'Integer', iptrw);
17354:     SelEnd', 'Integer', iptrw); AlwaysShowCaret', 'Boolean', iptrw);
17355:     Procedure UpdateCaret
17356:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2:word;SS2:TShiftState);
17357:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2: word;SS2:TShiftState);
17358:     Procedure BeginUndoBlock
17359:     Procedure BeginUpdate
17360:     Function CaretInView : Boolean
17361:     Function CharIndexToRowCol( Index : integer) : TBufferCoord
17362:     Procedure Clear
17363:     Procedure ClearAll
17364:     Procedure ClearBookMark( BookMark : Integer)
17365:     Procedure ClearSelection
17366:     Procedure CommandProcessor( Command : TSynEditorCommand; AChar : char; Data : pointer)
17367:     Procedure ClearUndo
17368:     Procedure CopyToClipboard
17369:     Procedure CutToClipboard
17370:     Procedure DoCopyToClipboard( const SText : string)
17371:     Procedure EndUndoBlock
17372:     Procedure EndUpdate
17373:     Procedure EnsureCursorPosVisible
17374:     Procedure EnsureCursorPosVisibleEx( ForceToMiddle : Boolean)
17375:     Procedure FindMatchingBracket
17376:     Function GetMatchingBracket : TBufferCoord
17377:     Function GetMatchingBracketEx( const APoint : TBufferCoord) : TBufferCoord
17378:     Procedure ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer)
17379:     Function GetBookMark( BookMark : integer; var X, Y : integer) : boolean
17380:     Function GetHighlighterAttriAtRowCol( const XY: TBufferCoord; var Token : string; var Attri
17381:               : TSynHighlighterAttributes) : boolean
```

```
17382:     Function GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token : string;
17383:              var TokenType, Start : Integer; var Attri:TSynHighlighterAttributes):boolean
17384:     Function GetPositionOfMouse( out aPos : TBufferCoord) : Boolean
17385:     Function GetWordAtRowCol( const XY : TBufferCoord) : string
17386:     Procedure GotoBookMark( BookMark : Integer)
17387:     Procedure GotoLineAndCenter( ALine : Integer)
17388:     Function IdentChars : TSynIdentChars
17389:     Procedure InvalidateGutter
17390:     Procedure InvalidateGutterLine( aLine : integer)
17391:     Procedure InvalidateGutterLines( FirstLine, LastLine : integer)
17392:     Procedure InvalidateLine( Line : integer)
17393:     Procedure InvalidateLines( FirstLine, LastLine : integer)
17394:     Procedure InvalidateSelection
17395:     Function IsBookmark( BookMark : integer) : boolean
17396:     Function IsPointInSelection( const Value : TBufferCoord) : boolean
17397:     Procedure LockUndo
17398:     Function BufferToDisplayPos( const p : TBufferCoord) : TDisplayCoord
17399:     Function DisplayToBufferPos( const p : TDisplayCoord) : TBufferCoord
17400:     Function LineToRow( aLine : integer) : integer
17401:     Function RowToLine( aRow : integer) : integer
17402:     Function NextWordPos : TBufferCoord
17403:     Function NextWordPosEx( const XY : TBufferCoord) : TBufferCoord
17404:     Procedure PasteFromClipboard
17405:     Function WordStart : TBufferCoord
17406:     Function WordStartEx( const XY : TBufferCoord) : TBufferCoord
17407:     Function WordEnd : TBufferCoord
17408:     Function WordEndEx( const XY : TBufferCoord) : TBufferCoord
17409:     Function PrevWordPos : TBufferCoord
17410:     Function PrevWordPosEx( const XY : TBufferCoord) : TBufferCoord
17411:     Function PixelsToRowColumn( aX, aY : integer) : TDisplayCoord
17412:     Function PixelsToNearestRowColumn( aX, aY : integer) : TDisplayCoord
17413:     Procedure Redo
17414:     Procedure RegisterCommandHandler(const AHandlerProc:THookedCommandEvent;AHandlerData:pointer));
17415:     Function RowColumnToPixels( const RowCol : TDisplayCoord) : TPoint
17416:     Function RowColToCharIndex( RowCol : TBufferCoord) : integer
17417:     Function SearchReplace( const ASearch, AReplace:string; AOptions:TSynSearchOptions): integer
17418:     Procedure SelectAll
17419:     Procedure SetBookMark( BookMark : Integer; X : Integer; Y : Integer)
17420:     Procedure SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord)
17421:     Procedure SetDefaultKeystrokes
17422:     Procedure SetSelWord
17423:     Procedure SetWordBlock( Value : TBufferCoord)
17424:     Procedure Undo
17425:     Procedure UnlockUndo
17426:     Procedure UnregisterCommandHandler( AHandlerProc : THookedCommandEvent)
17427:     Procedure AddKeyUpHandler( aHandler : TKeyEvent)
17428:     Procedure RemoveKeyUpHandler( aHandler : TKeyEvent)
17429:     Procedure AddKeyDownHandler( aHandler : TKeyEvent)
17430:     Procedure RemoveKeyDownHandler( aHandler : TKeyEvent)
17431:     Procedure AddKeyPressHandler( aHandler : TKeyPressEvent)
17432:     Procedure RemoveKeyPressHandler( aHandler : TKeyPressEvent)
17433:     Procedure AddFocusControl( aControl : TWinControl)
17434:     Procedure RemoveFocusControl( aControl : TWinControl)
17435:     Procedure AddMouseDownHandler( aHandler : TMouseEvent)
17436:     Procedure RemoveMouseDownHandler( aHandler : TMouseEvent)
17437:     Procedure AddMouseUpHandler( aHandler : TMouseEvent)
17438:     Procedure RemoveMouseUpHandler( aHandler : TMouseEvent)
17439:     Procedure AddMouseCursorHandler( aHandler : TMouseCursorEvent)
17440:     Procedure RemoveMouseCursorHandler( aHandler : TMouseCursorEvent)
17441:     Procedure SetLinesPointer( ASynEdit : TCustomSynEdit)
17442:     Procedure RemoveLinesPointer
17443:     Procedure HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList)
17444:     Procedure UnHookTextBuffer
17445:     BlockBegin', 'TBufferCoord', iptrw);
17446:     BlockEnd', 'TBufferCoord', iptrw);
17447:     CanPaste', 'Boolean', iptr);
17448:     CanRedo', 'boolean', iptr);
17449:     CanUndo', 'boolean', iptr);
17450:     CaretX', 'Integer', iptrw);
17451:     CaretY', 'Integer', iptrw);
17452:     CaretXY', 'TBufferCoord', iptrw);
17453:     ActiveLineColor', 'TColor', iptrw);
17454:     DisplayX', 'Integer', iptr);
17455:     DisplayY', 'Integer', iptr);
17456:     DisplayXY', 'TDisplayCoord', iptr);
17457:     DisplayLineCount', 'integer', iptr);
17458:     CharsInWindow', 'Integer', iptr);
17459:     CharWidth', 'integer', iptr);
17460:     Font', 'TFont', iptrw);
17461:     GutterWidth', 'Integer', iptr);
17462:     Highlighter', 'TSynCustomHighlighter', iptrw);
17463:     LeftChar', 'Integer', iptrw);
17464:     LineHeight', 'integer', iptr);
17465:     LinesInWindow', 'Integer', iptr);
17466:     LineText', 'string', iptrw);
17467:     Lines', 'TStrings', iptrw);
17468:     Marks', 'TSynEditMarkList', iptr);
17469:     MaxScrollWidth', 'integer', iptrw);
17470:     Modified', 'Boolean', iptrw);
```

```
17471:     PaintLock', 'Integer', iptr);
17472:     ReadOnly', 'Boolean', iptrw);
17473:     SearchEngine', 'TSynEditSearchCustom', iptrw);
17474:     SelAvail', 'Boolean', iptr); SelLength', 'integer', iptrw);
17475:     SelTabBlock', 'Boolean', iptr);
17476:     SelTabLine', 'Boolean', iptr);
17477:     SelText', 'string', iptrw);
17478:     StateFlags', 'TSynStateFlags', iptr);
17479:     Text', 'string', iptrw);
17480:     TopLine', 'Integer', iptrw);
17481:     WordAtCursor', 'string', iptr);
17482:     WordAtMouse', 'string', iptr);
17483:     UndoList', 'TSynEditUndoList', iptr);
17484:     RedoList', 'TSynEditUndoList', iptr);
17485:     OnProcessCommand', 'TProcessCommandEvent', iptrw);
17486:     BookMarkOptions', 'TSynBookMarkOpt', iptrw);
17487:     BorderStyle', 'TSynBorderStyle', iptrw);
17488:     ExtraLineSpacing', 'integer', iptrw);
17489:     Gutter', 'TSynGutter', iptrw);
17490:     HideSelection', 'boolean', iptrw);
17491:     InsertCaret', 'TSynEditCaretType', iptrw);
17492:     InsertMode', 'boolean', iptrw);
17493:     IsScrolling', 'Boolean', iptr);
17494:     Keystrokes', 'TSynEditKeyStrokes', iptrw);
17495:     MaxUndo', 'Integer', iptrw);
17496:     Options', 'TSynEditorOptions', iptrw);
17497:     OverwriteCaret', 'TSynEditCaretType', iptrw);
17498:     RightEdge', 'Integer', iptrw); RightEdgeColor', 'TColor', iptrw);
17499:     ScrollHintColor', 'TColor', iptrw);
17500:     ScrollHintFormat', 'TScrollHintFormat', iptrw);
17501:     ScrollBars', 'TScrollStyle', iptrw);
17502:     SelectedColor', 'TSynSelectedColor', iptrw);
17503:     SelectionMode', 'TSynSelectionMode', iptrw);
17504:     ActiveSelectionMode', 'TSynSelectionMode', iptrw);
17505:     TabWidth', 'integer', iptrw);  WantReturns', 'boolean', iptrw);
17506:     WantTabs', 'boolean', iptrw);  WordWrap', 'boolean', iptrw);
17507:     WordWrapGlyph', 'TSynGlyph', iptrw);
17508:     OnChange', 'TNotifyEvent', iptrw);
17509:     OnClearBookmark', 'TPlaceMarkEvent', iptrw);
17510:     OnCommandProcessed', 'TProcessCommandEvent', iptrw);
17511:     OnContextHelp', 'TContextHelpEvent', iptrw);
17512:     OnDropFiles', 'TDropFilesEvent', iptrw);
17513:     OnGutterClick', 'TGutterClickEvent', iptrw);
17514:     OnGutterGetText', 'TGutterGetTextEvent', iptrw);
17515:     OnGutterPaint', 'TGutterPaintEvent', iptrw);
17516:     OnMouseCursor', 'TMouseCursorEvent', iptrw);
17517:     OnPaint', 'TPaintEvent', iptrw);
17518:     OnPlaceBookmark', 'TPlaceMarkEvent', iptrw);
17519:     OnProcessUserCommand', 'TProcessCommandEvent', iptrw);
17520:     OnReplaceText', 'TReplaceTextEvent', iptrw);
17521:     OnSpecialLineColors', 'TSpecialLineColorsEvent', iptrw);
17522:     OnStatusChange', 'TStatusChangeEvent', iptrw);
17523:     OnPaintTransient', 'TPaintTransient', iptrw);
17524:     OnScroll', 'TScrollEvent', iptrw);
17525:    end;
17526:  end;
17527:  Procedure RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)
17528:  Function GetPlaceableHighlighters : TSynHighlighterList
17529:  Function EditorCommandToDescrString( Cmd : TSynEditorCommand) : string
17530:  Function EditorCommandToCodeString( Cmd : TSynEditorCommand) : string
17531:  Procedure GetEditorCommandValues( Proc : TGetStrProc)
17532:  Procedure GetEditorCommandExtended( Proc : TGetStrProc)
17533:  Function IdentToEditorCommand( const Ident : string; var Cmd : longint) : boolean
17534:  Function EditorCommandToIdent( Cmd : longint; var Ident : string) : boolean
17535:  Function ConvertCodeStringToExtended( AString : String) : String
17536:  Function ConvertExtendedToCodeString( AString : String) : String
17537:  Function ConvertExtendedToCommand( AString : String) : TSynEditorCommand
17538:  Function ConvertCodeStringToCommand( AString : String) : TSynEditorCommand
17539:  Function IndexToEditorCommand( const AIndex : Integer) : Integer
17540:
17541:   TSynEditorOption = (
17542:     eoAltSetsColumnMode,       //Holding down the Alt Key will put the selection mode into columnar format
17543:     eoAutoIndent,              //Will indent caret on newlines with same amount of leading whitespace as
17544:                                // preceding line
17545:     eoAutoSizeMaxScrollWidth,  //Automatically resizes the MaxScrollWidth property when inserting text
17546:     eoDisableScrollArrows,     //Disables the scroll bar arrow buttons when you can't scroll in that
17547:                                //direction any more
17548:     eoDragDropEditing,         //Allows to select a block of text and drag it within document to another
17549:                                // location
17550:     eoDropFiles,               //Allows the editor accept OLE file drops
17551:     eoEnhanceHomeKey,          //enhances home key positioning, similar to visual studio
17552:     eoEnhanceEndKey,           //enhances End key positioning, similar to JDeveloper
17553:     eoGroupUndo,               //When undoing/redoing actions, handle all continous changes the same kind
17554:                                // in one call
17555:                                //instead undoing/redoing each command separately
17556:     eoHalfPageScroll,          //When scrolling with page-up and page-down commands, only scroll a half
17557:                                //page at a time
17558:     eoHideShowScrollbars,      //if enabled, then scrollbars will only show if necessary.
17559:     If you have ScrollPastEOL, then it the horizontal bar will always be there (it uses MaxLength instead)
```

```
17560:     eoKeepCaretX,              //When moving through lines w/o cursor Past EOL, keeps X position of cursor
17561:     eoNoCaret,                 //Makes it so the caret is never visible
17562:     eoNoSelection,             //Disables selecting text
17563:     eoRightMouseMovesCursor,   //When clicking with right mouse for popup menu, moves cursor to location
17564:     eoScrollByOneLess,         //Forces scrolling to be one less
17565:     eoScrollHintFollows,       //The scroll hint follows the mouse when scrolling vertically
17566:     eoScrollPastEof,           //Allows the cursor to go past the end of file marker
17567:     eoScrollPastEol,           //Allows cursor to go past last character into white space at end of a line
17568:     eoShowScrollHint,          //Shows a hint of the visible line numbers when scrolling vertically
17569:     eoShowSpecialChars,        //Shows the special Characters
17570:     eoSmartTabDelete,          //similar to Smart Tabs, but when you delete characters
17571:     eoSmartTabs,               //When tabbing, cursor will go to non-white space character of previous line
17572:     eoSpecialLineDefaultFg,    //disables the foreground text color override using OnSpecialLineColor event
17573:     eoTabIndent,               //If active <Tab>and<Shift><Tab> act block indent,unindent when text select
17574:     eoTabsToSpaces,            //Converts a tab character to a specified number of space characters
17575:     eoTrimTrailingSpaces       //Spaces at the end of lines will be trimmed and not saved
17576:
17577:     ********************************Important Editor Short Cuts*****************************);
17578:     Double click to select a word and count words with highlightning.
17579:     Triple click to select a line.
17580:     CTRL+SHIFT+click to extend a selection.
17581:     Drag with the ALT key down to select columns of text !!!
17582:     Drag and drop is supported.
17583:     Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
17584:     Type CTRL+A to select all.
17585:     Type CTRL+N to set a new line.
17586:     Type CTRL+T to delete a line or token. //Tokenizer
17587:     Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
17588:     Type CTRL+Shift+T to add ToDo in line and list.
17589:     Type CTRL+Shift+[0..9] to set bookmarks.   //Bookmark
17590:     Type CTRL[0..9] to jump or get to bookmarks.
17591:     Type Home to position cursor at beginning of current line and End to position it at end of line.
17592:     Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of document.
17593:     Page Up and Page Down work as expected.
17594:     CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
17595:     using http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
17596:
17597: {$ Short Key Positions Ctrl<A-Z>: }
17598: def
17599:    <A> Select All
17600:    <B> Count Words
17601:    <C> Copy
17602:    <D> Internet Start
17603:    <E> Script List
17604:    <F> Find
17605:    <G> Goto
17606:    <H> Mark Line
17607:    <I> Interface List
17608:    <J> Code Completion
17609:    <K> Console
17610:    <L> Interface List Box
17611:    <M> Font Larger -
17612:    <N> New Line
17613:    <O> Open File
17614:    <P> Font Smaller +
17615:    <Q> Quit
17616:    <R> Replace
17617:    <S> Save!
17618:    <T> Delete Line
17619:    <U> Use Case Editor
17620:    <V> Paste
17621:    <W> URI Links
17622:    <X> Reserved for coding use internal
17623:    <Y> Delete Line
17624:    <Z> Undo
17625:
17626: ref
17627:    F1 Help
17628:    F2 Syntax Check
17629:    F3 Search Next
17630:    F4 New Instance
17631:    F5 Line Mark /Breakpoint
17632:    F6 Goto End
17633:    F7 Debug Step Into
17634:    F8 Debug Step Out
17635:    F9 Compile
17636:    F10 Menu
17637:    F11 Word Count Highlight
17638:    F12 Reserved for coding use internal
17639:
17640:  def ReservedWords: array[0..78] of string =
17641:     ('and', 'array', 'as', 'asm', 'at', 'begin', 'case', 'class', 'const',
17642:      'constructor', 'default', 'destructor', 'dispinterface', 'div', 'do',
17643:      'downto', 'else', 'end', 'except', 'exports', 'file', 'finalization',
17644:      'finally', 'for', 'function', 'goto', 'if', 'implementation', 'in',
17645:      'inherited', 'initialization', 'inline', 'interface', 'is', 'label',
17646:      'library', 'message', 'mod', 'nil', 'not', 'object', 'of', 'on', 'or',
17647:      'out', 'packed', 'procedure', 'program', 'property', 'raise', 'read',
17648:      'record', 'repeat', 'resourcestring', 'set', 'shl', 'shr', 'string',
```

```
17649:      'stored', 'then', 'threadvar', 'to', 'try', 'type', 'unit', 'until',
17650:      'uses', 'var', 'while', 'with', 'write', 'xor', 'private', 'protected',
17651:      'public', 'published','def','ref','using
17652:   AllowedChars: array[0..5] of string = ('(',')', '[', ']',' ',' t,t1,t2,t3: boolean;
17653:
17654: //-------------------------------------------------------------------------
17655: //*************End of mX4 Public  Tools API *********************************
17656: //-------------------------------------------------------------------------
17657:
17658: Amount of Functions: 11299
17659: Amount of Procedures: 7289
17660: Amount of Constructors: 1181
17661: Totals of Calls: 19769
17662: SHA1: Win 3.9.9.88 119533C0725A9B9B2919849759AA2F6298EBFF28
17663:
17664:
17665: ************************************************************
17666:  Short Manual with 50 Tips!
17667: ************************************************************
17668: - Install: just save your maxboxdef.ini before and then extract the zip file!
17669: - Toolbar: Click on the red maXbox Sign (right on top) opens your work directory or jump to <Help>
17670: - Menu: With <F2> you check syntax with <F8> you debug and <F9> you compile!
17671: - Menu: With <Crtl><F3> you can search for code on examples
17672: - Menu: Open in menu Output a new instance <F4> of the box to compare or prepare your scripts
17673: - Menu: Set Interface Naviagator in menu /View/Intf Navigator
17674: - Menu: Switch or toogle between the last 2 scripts in menu File/LoadLast (History is set to 9 files)
17675:
17676: - Inifile: Set memory report in ini: MEMORYREPORT=Y :report on memory leaks on shutdown by dialog
17677: - Inifile: Refresh (reload) the inifile after edit with ../Help/Config Update
17678: - Context Menu: You can printout your scripts as a pdf-file or html-export
17679: - Context: You do have a context menu with the right mouse click
17680:
17681: - Menu: With the UseCase Editor you can convert graphic formats too.
17682: - Menu: On menu Options you find Addons as compiled scripts
17683: - IDE: You don't need a mouse to handle maXbox, use shortcuts
17684: - Menu: Check Options/ProcessMessages! if something is wrong or you can't see graphics in a time
17685: - IDE: Dragndrop your scripts in box or the model in use case editor (Cut,Copy,Paste always available)
17686: - Editor: You can get templates as code completion with <ctrl j> in editor like classp or iinterface
17687:          or ttimer (you type classp and then CTRL J),or you type tstringlist and <Ctrl><J>
17688:
17689: - Menu: In menu output (console) you can set output menu in edit mode by unchecking <read only output>
17690: - Editor: After the end. you can write or copy notes or descriptions concerning the app or code
17691: - Code: If you code a loop till key-pressed use function: isKeyPressed;
17692: - Code: Macro set the macros #name, #date, #host, #path, #file, #head #sign, Tutorial maxbox_starter25.pdf
17693: - Code: change Syntax in autoboot macro 'maxbootscript.txt'
17694: - Editor: - <F11> Click on Word in Editor search amount of words with highlighting, Dbl Click on Bookmarks
17695:          to delete and Click and mark to drag a bookmark
17696: - Menu: To start handling from CD-ROM (read only mode) uncheck in Menu /Options/Save before Compile
17697: - IDE: A file info with system and script information you find in menu Program/Information
17698: - IDE: After change the config file in help you can update changes in menu Help/Config Update
17699: - IDE: Make a screenshot of the content and environment in menu Output/Save Screenshot
17700: - IDE: Use a boot loader script 'maxbootscript.txt' (as auto start) to change box each time you start it.
17701: - IDE: With escape or <Ctrl> Q you can also leave the box or stop a script in menu program - stop program
17702: - Editor: Set Bookmarks to check your work in app or code
17703: - Editor: With <Ctrl H> you set {$Active Line Color} and F11 you get Word Count Statistic on Output too
17704: - Editor: With {///TODO: some description} or DONE you set code entries for ToDo List in ../Help/ToDo List
17705: - Editor: With <Ctrl W> you set active URL links in your code to test availability in Context Menu
17706:
17707: - IDE with  menu /Options/ADO SQL Workbench you can manage your Database
17708: - Context Menu: You can write your docus with RichEdit RTF printout /Editor Form Options/Richedit
17709: - Menu: Set Interface Naviagator also with  toogle <Ctrl L> or /View/Intf Navigator
17710: - Toolbar: In menu /View switch Off Toolbar and Coolbar to get more coding space
17711: - Code: Put some resources in your code /Help/Resource Explorer like bitbtn, forms, dialogs;
17712: - Code Editor: Compile with <F9> but also Alt C in case <F9> isnt available;
17713: - IDE set bookmarks with  <Ctrl Shift> (0..9) and jump with <Ctrl> (0..9)
17714: - IDE menu /Help/Tools/ write with RTF Editor or open a DOS Shell or browse
17715:
17716: - Add on write your Requirements in RTF Docu with <CTRL ALT R> in context menu
17717: - Add on when no browser is available start /Options/Add ons/Easy Browser
17718: - Add on SOAP Tester with SOP POST File
17719: - Add on IP Protocol Sniffer with List View
17720:
17721: - Menu: Help/Tools as a Tool Section with DOS Opener
17722: - Menu Editor: export the code as RTF File
17723: - Menu: Help/Tools/Syn Export your code as available in HTML or RTF
17724: - Menu: Help/Tools/ with <Alt H-S-D> you start the DOS Shell
17725: - Context: Auto Detect of Syntax depending on file extension
17726: - Code: some Windows API function start with w in the name like wGetAtomName();
17727: - IDE Close - if you cant close the box then reset it <Ctrl F2> in menu /Debug
17728:
17729: - using DLL example in maXbox: //function: {*****************************************}
17730:   Function GetProcessMemoryInfo(Process: THandle; var MemoryCounters: TProcessMemoryCounters;
17731:                                 cb: DWORD): BOOL; //stdcall;;
17732:      External 'GetProcessMemoryInfo@psapi.dll stdcall';
17733:
17734:   Function OpenProcess(dwDesiredAccess:DWORD; bInheritHandle:BOOL; dwProcessId: DWORD):THandle;
17735:      External  'OpenProcess@kernel32.dll stdcall';
17736:
17737: PCT Precompile Technology , mX4 ScriptStudio
```

```
17738: Indy, JCL, Jedi, VCL, Systools, TurboPower, Fundamentals, ExtendedRTL, Synedit
17739: DMath, devC, Graphics32, ExtPascal, mX4, LCL, CLX, FCL, CPort and more
17740: emax layers: system-package-component-unit-class-function-block
17741: new keywords def ref using maXCalcF
17742: UML: use case act class state seq pac comp dep - lib lab
17743: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
17744: Tutorials, 30 Units add, VCL constructors, controls plus, unit list
17745:
17746:
17747:
17748: ********************************************************
17749:  unit List asm internal end
17750: ********************************************************
17751: 01 unit RIRegister_StrUtils_Routines(exec);    //Delphi
17752: 02 unit SIRegister_IdStrings                   //Indy Sockets
17753: 03 unit RIRegister_niSTRING_Routines(Exec);    //from RegEx
17754: 04 unit uPSI_fMain Functions;                  //maXbox Open Tools API
17755: 05 unit IFSI_WinForm1puzzle;                   //maXbox
17756: 06 unit RIRegister_LinarBitmap_Routines(Exec); //ImageFileLibBCB
17757: 07 unit RegisterDateTimeLibrary_R(exec);      //Delphi
17758: 08 unit RIRegister_MathMax_Routines(exec);    //Jedi & Delphi
17759: 09 unit RIRegister_IdGlobal_Routines(exec);   //Indy Sockets
17760: 10 unit RIRegister_SysUtils_Routines(Exec);   //Delphi
17761: 11 unit uPSI_IdTCPConnection;                 //Indy some functions
17762: 12 unit uPSCompiler.pas;                      //PS kernel functions
17763: 13 unit uPSI_DBCommon;                        //DB Common_Routines and Types
17764: 14 unit uPSI_Printers.pas                     //Delphi VCL
17765: 15 unit uPSI_MPlayer.pas                     //Delphi VCL
17766: 16 unit uPSC_comobj;                         //COM Functions
17767: 17 unit uPSI_Clipbrd;                        //Delphi VCL
17768: 18 unit Filectrl in IFSI_SysUtils_max;      //VCL Runtime
17769: 19 unit uPSI_SqlExpr;                       //DBX3
17770: 20 unit uPSI_ADODB;                         //ADODB
17771: 21 unit uPSI_StrHlpr;                       //String Helper Routines
17772: 22 unit uPSI_DateUtils;                     //Expansion to DateTimeLib
17773: 23 unit uPSI_FileUtils;                     //Expansion to Sys/File Utils
17774: 24 unit JUtils / gsUtils;                   //Jedi / Metabase
17775: 25 unit JvFunctions_max;                    //Jedi Functions
17776: 26 unit HTTPParser;                         //Delphi VCL
17777: 27 unit HTTPUtil;                           //Delphi VCL
17778: 28 unit uPSI_XMLUtil;                       //Delphi VCL
17779: 29 unit uPSI_SOAPHTTPClient;                //Delphi VCL SOAP WebService V3.5
17780: 30 unit uPSI_Contnrs;                       //Delphi RTL Container of Classes
17781: 31 unit uPSI_MaskUtils;                     //RTL Edit and Mask functions
17782: 32 unit uPSI_MyBigInt;                      //big integer class with Math
17783: 33 unit uPSI_ConvUtils;                     //Delphi VCL Conversions engine
17784: 34 unit Types_Variants;                     //Delphi\Win32\rtl\sys
17785: 35 unit uPSI_IdHashSHA1;                    //Indy Crypto Lib
17786: 36 unit uPSI_IdHashMessageDigest           //Indy Crypto;
17787: 37 unit uPSI_IdASN1Util;                    //Indy ASN1Utility Routines;
17788: 38 unit uPSI_IdLogFile;                     //Indy Logger from LogBase
17789: 39 unit uPSI_IdIcmpClient;                  //Indy Ping ICMP
17790: 40 unit uPSI_IdHashMessageDigest_max       //Indy Crypto &OpenSSL;
17791: 41 unit uPSI_FileCtrl;                      //Delphi RTL
17792: 42 unit uPSI_Outline;                       //Delphi VCL
17793: 43 unit uPSI_ScktComp;                      //Delphi RTL
17794: 44 unit uPSI_Calendar;                      //Delphi VCL
17795: 45 unit uPSI_VListView                      //VListView;
17796: 46 unit uPSI_DBGrids;                       //Delphi VCL
17797: 47 unit uPSI_DBCtrls;                       //Delphi VCL
17798: 48 unit ide_debugoutput;                    //maXbox
17799: 49 unit uPSI_ComCtrls;                     //Delphi VCL
17800: 50 unit uPSC_stdctrls+;                    //Delphi VCL
17801: 51 unit uPSI_Dialogs;                       //Delphi VCL
17802: 52 unit uPSI_StdConvs;                     //Delphi RTL
17803: 53 unit uPSI_DBClient;                     //Delphi RTL
17804: 54 unit uPSI_DBPlatform;                   //Delphi RTL
17805: 55 unit uPSI_Provider;                     //Delphi RTL
17806: 56 unit uPSI_FMTBcd;                       //Delphi RTL
17807: 57 unit uPSI_DBCGrids;                     //Delphi VCL
17808: 58 unit uPSI_CDSUtil;                      //MIDAS
17809: 59 unit uPSI_VarHlpr;                      //Delphi RTL
17810: 60 unit uPSI_ExtDlgs;                      //Delphi VCL
17811: 61 unit sdpStopwatch;                      //maXbox
17812: 62 unit uPSI_JclStatistics;                //JCL
17813: 63 unit uPSI_JclLogic;                     //JCL
17814: 64 unit uPSI_JclMiscel;                    //JCL
17815: 65 unit uPSI_JclMath_max;                  //JCL RTL
17816: 66 unit uPSI_uTPLb_StreamUtils;            //LockBox 3
17817: 67 unit uPSI_MathUtils;                    //BCB
17818: 68 unit uPSI_JclMultimedia;                //JCL
17819: 69 unit uPSI_WideStrUtils;                 //Delphi API/RTL
17820: 70 unit uPSI_GraphUtil;                    //Delphi RTL
17821: 71 unit uPSI_TypeTrans;                    //Delphi RTL
17822: 72 unit uPSI_HTTPApp;                      //Delphi VCL
17823: 73 unit uPSI_DBWeb;                        //Delphi VCL
17824: 74 unit uPSI_DBBdeWeb;                     //Delphi VCL
17825: 75 unit uPSI_DBXpressWeb;                  //Delphi VCL
17826: 76 unit uPSI_ShadowWnd;                    //Delphi VCL
```

```
17827: 77 unit uPSI_ToolWin;                        //Delphi VCL
17828: 78 unit uPSI_Tabs;                           //Delphi VCL
17829: 79 unit uPSI_JclGraphUtils;                  //JCL
17830: 80 unit uPSI_JclCounter;                     //JCL
17831: 81 unit uPSI_JclSysInfo;                     //JCL
17832: 82 unit uPSI_JclSecurity;                    //JCL
17833: 83 unit uPSI_JclFileUtils;                   //JCL
17834: 84 unit uPSI_IdUserAccounts;                 //Indy
17835: 85 unit uPSI_IdAuthentication;               //Indy
17836: 86 unit uPSI_uTPLb_AES;                      //LockBox 3
17837: 87 unit uPSI_IdHashSHA1;                     //LockBox 3
17838: 88 unit uTPLb_BlockCipher;                   //LockBox 3
17839: 89 unit uPSI_ValEdit.pas;                    //Delphi VCL
17840: 90 unit uPSI_JvVCLUtils;                     //JCL
17841: 91 unit uPSI_JvDBUtil;                       //JCL
17842: 92 unit uPSI_JvDBUtils;                      //JCL
17843: 93 unit uPSI_JvAppUtils;                     //JCL
17844: 94 unit uPSI_JvCtrlUtils;                    //JCL
17845: 95 unit uPSI_JvFormToHtml;                   //JCL
17846: 96 unit uPSI_JvParsing;                      //JCL
17847: 97 unit uPSI_SerDlgs;                        //Toolbox
17848: 98 unit uPSI_Serial;                         //Toolbox
17849: 99 unit uPSI_JvComponent;                    //JCL
17850: 100 unit uPSI_JvCalc;                       //JCL
17851: 101 unit uPSI_JvBdeUtils;                   //JCL
17852: 102 unit uPSI_JvDateUtil;                   //JCL
17853: 103 unit uPSI_JvGenetic;                    //JCL
17854: 104 unit uPSI_JclBase;                      //JCL
17855: 105 unit uPSI_JvUtils;                      //JCL
17856: 106 unit uPSI_JvStrUtil;                    //JCL
17857: 107 unit uPSI_JvStrUtils;                   //JCL
17858: 108 unit uPSI_JvFileUtil;                   //JCL
17859: 109 unit uPSI_JvMemoryInfos;                //JCL
17860: 110 unit uPSI_JvComputerInfo;               //JCL
17861: 111 unit uPSI_JvgCommClasses;               //JCL
17862: 112 unit uPSI_JvgLogics;                    //JCL
17863: 113 unit uPSI_JvLED;                        //JCL
17864: 114 unit uPSI_JvTurtle;                     //JCL
17865: 115 unit uPSI_SortThds; unit uPSI_ThSort;   //maXbox
17866: 116 unit uPSI_JvgUtils;                     //JCL
17867: 117 unit uPSI_JvExprParser;                 //JCL
17868: 118 unit uPSI_HexDump;                      //Borland
17869: 119 unit uPSI_DBLogDlg;                     //VCL
17870: 120 unit uPSI_SqlTimSt;                     //RTL
17871: 121 unit uPSI_JvHtmlParser;                 //JCL
17872: 122 unit uPSI_JvgXMLSerializer;             //JCL
17873: 123 unit uPSI_JvJCLUtils;                   //JCL
17874: 124 unit uPSI_JvStrings;                    //JCL
17875: 125 unit uPSI_uTPLb_IntegerUtils;           //TurboPower
17876: 126 unit uPSI_uTPLb_HugeCardinal;           //TurboPower
17877: 127 unit uPSI_uTPLb_HugeCardinalUtils;      //TurboPower
17878: 128 unit uPSI_SynRegExpr;                   //SynEdit
17879: 129 unit uPSI_StUtils;                      //SysTools4
17880: 130 unit uPSI_StToHTML;                     //SysTools4
17881: 131 unit uPSI_StStrms;                      //SysTools4
17882: 132 unit uPSI_StFIN;                        //SysTools4
17883: 133 unit uPSI_StAstroP;                     //SysTools4
17884: 134 unit uPSI_StStat;                       //SysTools4
17885: 135 unit uPSI_StNetCon;                     //SysTools4
17886: 136 unit uPSI_StDecMth;                     //SysTools4
17887: 137 unit uPSI_StOStr;                       //SysTools4
17888: 138 unit uPSI_StPtrns;                      //SysTools4
17889: 139 unit uPSI_StNetMsg;                     //SysTools4
17890: 140 unit uPSI_StMath;                       //SysTools4
17891: 141 unit uPSI_StExpEng;                     //SysTools4
17892: 142 unit uPSI_StCRC;                        //SysTools4
17893: 143 unit uPSI_StExport,                     //SysTools4
17894: 144 unit uPSI_StExpLog,                     //SysTools4
17895: 145 unit uPSI_ActnList;                     //Delphi VCL
17896: 146 unit uPSI_jpeg;                         //Borland
17897: 147 unit uPSI_StRandom;                     //SysTools4
17898: 148 unit uPSI_StDict;                       //SysTools4
17899: 149 unit uPSI_StBCD;                        //SysTools4
17900: 150 unit uPSI_StTxtDat;                     //SysTools4
17901: 151 unit uPSI_StRegEx;                      //SysTools4
17902: 152 unit uPSI_IMouse;                       //VCL
17903: 153 unit uPSI_SyncObjs;                     //VCL
17904: 154 unit uPSI_AsyncCalls;                   //Hausladen
17905: 155 unit uPSI_ParallelJobs;                 //Saraiva
17906: 156 unit uPSI_Variants;                     //VCL
17907: 157 unit uPSI_VarCmplx;                     //VCL Wolfram
17908: 158 unit uPSI_DTDSchema;                    //VCL
17909: 159 unit uPSI_ShLwApi;                      //Brakel
17910: 160 unit uPSI_IBUtils;                      //VCL
17911: 161 unit uPSI_CheckLst;                     //VCL
17912: 162 unit uPSI_JvSimpleXml;                  //JCL
17913: 163 unit uPSI_JclSimpleXml;                 //JCL
17914: 164 unit uPSI_JvXmlDatabase;                //JCL
17915: 165 unit uPSI_JvMaxPixel;                   //JCL
```

```
17916: 166 unit uPSI_JvItemsSearchs;                //JCL
17917: 167 unit uPSI_StExpEng2;                    //SysTools4
17918: 168 unit uPSI_StGenLog;                     //SysTools4
17919: 169 unit uPSI_JvLogFile;                    //Jcl
17920: 170 unit uPSI_CPort;                        //ComPort Lib v4.11
17921: 171 unit uPSI_CPortCtl;                     //ComPort
17922: 172 unit uPSI_CPortEsc;                     //ComPort
17923: 173 unit BarCodeScaner;                     //ComPort
17924: 174 unit uPSI_JvGraph;                      //JCL
17925: 175 unit uPSI_JvComCtrls;                   //JCL
17926: 176 unit uPSI_GUITesting;                   //D Unit
17927: 177 unit uPSI_JvFindFiles;                  //JCL
17928: 178 unit uPSI_StSystem;                     //SysTools4
17929: 179 unit uPSI_JvKeyboardStates;             //JCL
17930: 180 unit uPSI_JvMail;                       //JCL
17931: 181 unit uPSI_JclConsole;                   //JCL
17932: 182 unit uPSI_JclLANMan;                    //JCL
17933: 183 unit uPSI_IdCustomHTTPServer;           //Indy
17934: 184 unit IdHTTPServer                       //Indy
17935: 185 unit uPSI_IdTCPServer;                  //Indy
17936: 186 unit uPSI_IdSocketHandle;               //Indy
17937: 187 unit uPSI_IdIOHandlerSocket;            //Indy
17938: 188 unit IdIOHandler;                       //Indy
17939: 189 unit uPSI_cutils;                       //Bloodshed
17940: 190 unit uPSI_BoldUtils;                    //boldsoft
17941: 191 unit uPSI_IdSimpleServer;               //Indy
17942: 192 unit uPSI_IdSSLOpenSSL;                 //Indy
17943: 193 unit uPSI_IdMultipartFormData;          //Indy
17944: 194 unit uPSI_SynURIOpener;                 //SynEdit
17945: 195 unit uPSI_PerlRegEx;                    //PCRE
17946: 196 unit uPSI_IdHeaderList;                 //Indy
17947: 197 unit uPSI_StFirst;                      //SysTools4
17948: 198 unit uPSI_JvCtrls;                      //JCL
17949: 199 unit uPSI_IdTrivialFTPBase;             //Indy
17950: 200 unit uPSI_IdTrivialFTP;                 //Indy
17951: 201 unit uPSI_IdUDPBase;                    //Indy
17952: 202 unit uPSI_IdUDPClient;                  //Indy
17953: 203 unit uPSI_utypes;                       //for DMath.DLL
17954: 204 unit uPSI_ShellAPI;                     //Borland
17955: 205 unit uPSI_IdRemoteCMDClient;            //Indy
17956: 206 unit uPSI_IdRemoteCMDServer;            //Indy
17957: 207 unit IdRexecServer;                     //Indy
17958: 208 unit IdRexec; (unit uPSI_IdRexec;)      //Indy
17959: 209 unit IdUDPServer;                       //Indy
17960: 210 unit IdTimeUDPServer;                   //Indy
17961: 211 unit IdTimeServer;                      //Indy
17962: 212 unit IdTimeUDP; (unit uPSI_IdUDPServer;) //Indy
17963: 213 unit uPSI_IdIPWatch;                    //Indy
17964: 214 unit uPSI_IdIrcServer;                  //Indy
17965: 215 unit uPSI_IdMessageCollection;          //Indy
17966: 216 unit uPSI_cPEM;                         //Fundamentals 4
17967: 217 unit uPSI_cFundamentUtils;              //Fundamentals 4
17968: 218 unit uPSI_uwinplot;                     //DMath
17969: 219 unit uPSI_xrtl_util_CPUUtils;           //ExtentedRTL
17970: 220 unit uPSI_GR32_System;                  //Graphics32
17971: 221 unit uPSI_cFileUtils;                   //Fundamentals 4
17972: 222 unit uPSI_cDateTime; (timemachine)      //Fundamentals 4
17973: 223 unit uPSI_cTimers; (high precision timer) //Fundamentals 4
17974: 224 unit uPSI_cRandom;                      //Fundamentals 4
17975: 225 unit uPSI_ueval;                        //DMath
17976: 226 unit uPSI_xrtl_net_URIUtils;            //ExtendedRTL
17977: 227 unit xrtl_net_URIUtils;                 //ExtendedRTL
17978: 228 unit uPSI_ufft;  (FFT)                  //DMath
17979: 229 unit uPSI_DBXChannel;                   //Delphi
17980: 230 unit uPSI_DBXIndyChannel;               //Delphi Indy
17981: 231 unit uPSI_xrtl_util_COMCat;             //ExtendedRTL
17982: 232 unit uPSI_xrtl_util_StrUtils;           //ExtendedRTL
17983: 233 unit uPSI_xrtl_util_VariantUtils;       //ExtendedRTL
17984: 234 unit uPSI_xrtl_util_FileUtils;          //ExtendedRTL
17985: 235 unit xrtl_util_Compat;                  //ExtendedRTL
17986: 236 unit uPSI_OleAuto;                      //Borland
17987: 237 unit uPSI_xrtl_util_COMUtils;           //ExtendedRTL
17988: 238 unit uPSI_CmAdmCtl;                     //Borland
17989: 239 unit uPSI_ValEdit2;                     //VCL
17990: 240 unit uPSI_GR32;  //Graphics32          //Graphics32
17991: 241 unit uPSI_GR32_Image;                   //Graphics32
17992: 242 unit uPSI_xrtl_util_TimeUtils;          //ExtendedRTL
17993: 243 unit uPSI_xrtl_util_TimeZone;           //ExtendedRTL
17994: 244 unit uPSI_xrtl_util_TimeStamp;          //ExtendedRTL
17995: 245 unit uPSI_xrtl_util_Map;                //ExtendedRTL
17996: 246 unit uPSI_xrtl_util_Set;                //ExtendedRTL
17997: 247 unit uPSI_CPortMonitor;                 //ComPort
17998: 248 unit uPSI_StIniStm;                     //SysTools4
17999: 249 unit uPSI_GR32_ExtImage;                //Graphics32
18000: 250 unit uPSI_GR32_OrdinalMaps;             //Graphics32
18001: 251 unit uPSI_GR32_Rasterizers;             //Graphics32
18002: 252 unit uPSI_xrtl_util_Exception;          //ExtendedRTL
18003: 253 unit uPSI_xrtl_util_Value;              //ExtendedRTL
18004: 254 unit uPSI_xrtl_util_Compare;            //ExtendedRTL
```

```
18005: 255 unit uPSI_FlatSB;                        //VCL
18006: 256 unit uPSI_JvAnalogClock;                 //JCL
18007: 257 unit uPSI_JvAlarms;                      //JCL
18008: 258 unit uPSI_JvSQLS;                        //JCL
18009: 259 unit uPSI_JvDBSecur;                     //JCL
18010: 260 unit uPSI_JvDBQBE;                       //JCL
18011: 261 unit uPSI_JvStarfield;                   //JCL
18012: 262 unit uPSI_JVCLMiscal;                    //JCL
18013: 263 unit uPSI_JvProfiler32;                  //JCL
18014: 264 unit uPSI_JvDirectories,                 //JCL
18015: 265 unit uPSI_JclSchedule,                   //JCL
18016: 266 unit uPSI_JclSvcCtrl,                    //JCL
18017: 267 unit uPSI_JvSoundControl,                //JCL
18018: 268 unit uPSI_JvBDESQLScript,                //JCL
18019: 269 unit uPSI_JvgDigits,                     //JCL>
18020: 270 unit uPSI_ImgList;                       //TCustomImageList
18021: 271 unit uPSI_JclMIDI;                       //JCL>
18022: 272 unit uPSI_JclWinMidi;                    //JCL>
18023: 273 unit uPSI_JclNTFS;                       //JCL>
18024: 274 unit uPSI_JclAppInst;                    //JCL>
18025: 275 unit uPSI_JvRle;                         //JCL>
18026: 276 unit uPSI_JvRas32;                       //JCL>
18027: 277 unit uPSI_JvImageDrawThread,             //JCL>
18028: 278 unit uPSI_JvImageWindow,                 //JCL>
18029: 279 unit uPSI_JvTransparentForm;             //JCL>
18030: 280 unit uPSI_JvWinDialogs;                  //JCL>
18031: 281 unit uPSI_JvSimLogic,                    //JCL>
18032: 282 unit uPSI_JvSimIndicator,                //JCL>
18033: 283 unit uPSI_JvSimPID,                      //JCL>
18034: 284 unit uPSI_JvSimPIDLinker,                //JCL>
18035: 285 unit uPSI_IdRFCReply;                    //Indy
18036: 286 unit uPSI_IdIdent;                       //Indy
18037: 287 unit uPSI_IdIdentServer;                 //Indy
18038: 288 unit uPSI_JvPatchFile;                   //JCL
18039: 289 unit uPSI_StNetPfm;                      //SysTools4
18040: 290 unit uPSI_StNet;                         //SysTools4
18041: 291 unit uPSI_JclPeImage;                    //JCL
18042: 292 unit uPSI_JclPrint;                      //JCL
18043: 293 unit uPSI_JclMime;                       //JCL
18044: 294 unit uPSI_JvRichEdit;                    //JCL
18045: 295 unit uPSI_JvDBRichEd;                    //JCL
18046: 296 unit uPSI_JvDice;                        //JCL
18047: 297 unit uPSI_JvFloatEdit;                   //JCL 3.9.8
18048: 298 unit uPSI_JvDirFrm;                      //JCL
18049: 299 unit uPSI_JvDualList;                    //JCL
18050: 300 unit uPSI_JvSwitch;                      ////JCL
18051: 301 unit uPSI_JvTimerLst;                    ////JCL
18052: 302 unit uPSI_JvMemTable;                    //JCL
18053: 303 unit uPSI_JvObjStr;                      //JCL
18054: 304 unit uPSI_StLArr;                        //SysTools4
18055: 305 unit uPSI_StWmDCpy;                      //SysTools4
18056: 306 unit uPSI_StText;                        //SysTools4
18057: 307 unit uPSI_StNTLog;                       //SysTools4
18058: 308 unit uPSI_xrtl_math_Integer;             //ExtendedRTL
18059: 309 unit uPSI_JvImagPrvw;                    //JCL
18060: 310 unit uPSI_JvFormPatch;                   //JCL
18061: 311 unit uPSI_JvPicClip;                     //JCL
18062: 312 unit uPSI_JvDataConv;                    //JCL
18063: 313 unit uPSI_JvCpuUsage;                    //JCL
18064: 314 unit uPSI_JclUnitConv_mX2;               //JCL
18065: 315 unit JvDualListForm;                     //JCL
18066: 316 unit uPSI_JvCpuUsage2;                   //JCL
18067: 317 unit uPSI_JvParserForm;                  //JCL
18068: 318 unit uPSI_JvJanTreeView;                 //JCL
18069: 319 unit uPSI_JvTransLED;                    //JCL
18070: 320 unit uPSI_JvPlaylist;                    //JCL
18071: 321 unit uPSI_JvFormAutoSize;                //JCL
18072: 322 unit uPSI_JvYearGridEditForm;            //JCL
18073: 323 unit uPSI_JvMarkupCommon;                //JCL
18074: 324 unit uPSI_JvChart;                       //JCL
18075: 325 unit uPSI_JvXPCore;                      //JCL
18076: 326 unit uPSI_JvXPCoreUtils;                 //JCL
18077: 327 unit uPSI_StatsClasses;                  //mX4
18078: 328 unit uPSI_ExtCtrls2;                     //VCL
18079: 329 unit uPSI_JvUrlGrabbers;                 //JCL
18080: 330 unit uPSI_JvXmlTree;                     //JCL
18081: 331 unit uPSI_JvWavePlayer;                  //JCL
18082: 332 unit uPSI_JvUnicodeCanvas;               //JCL
18083: 333 unit uPSI_JvTFUtils;                     //JCL
18084: 334 unit uPSI_IdServerIOHandler;             //Indy
18085: 335 unit uPSI_IdServerIOHandlerSocket;       //Indy
18086: 336 unit uPSI_IdMessageCoder;                //Indy
18087: 337 unit uPSI_IdMessageCoderMIME;            //Indy
18088: 338 unit uPSI_IdMIMETypes;                   //Indy
18089: 339 unit uPSI_JvConverter;                   //JCL
18090: 340 unit uPSI_JvCsvParse;                    //JCL
18091: 341 unit uPSI_umath;  unit uPSI_ugamma;      //DMath
18092: 342 unit uPSI_ExcelExport;(Nat:TJsExcelExport) //JCL
18093: 343 unit uPSI_JvDBGridExport;                //JCL
```

```
18094: 344 unit uPSI_JvgExport;                    //JCL
18095: 345 unit uPSI_JvSerialMaker;                //JCL
18096: 346 unit uPSI_JvWin32;                      //JCL
18097: 347 unit uPSI_JvPaintFX;                    //JCL
18098: 348 unit uPSI_JvOracleDataSet; (beta)       //JCL
18099: 349 unit uPSI_JvValidators; (preview)       //JCL
18100: 350 unit uPSI_JvNTEventLog;                 //JCL
18101: 351 unit uPSI_ShellZipTool;                 //mX4
18102: 352 unit uPSI_JvJoystick;                   //JCL
18103: 353 unit uPSI_JvMailSlots;                  //JCL
18104: 354 unit uPSI_JclComplex;                   //JCL
18105: 355 unit uPSI_SynPdf;                       //Synopse
18106: 356 unit uPSI_Registry;                     //VCL
18107: 357 unit uPSI_TlHelp32;                     //VCL
18108: 358 unit uPSI_JclRegistry;                  //JCL
18109: 359 unit uPSI_JvAirBrush;                   //JCL
18110: 360 unit uPSI_mORMotReport;                 //Synopse
18111: 361 unit uPSI_JclLocales;                   //JCL
18112: 362 unit uPSI_SynEdit;                      //SynEdit
18113: 363 unit uPSI_SynEditTypes;                 //SynEdit
18114: 364 unit uPSI_SynMacroRecorder;             //SynEdit
18115: 365 unit uPSI_LongIntList;                  //SynEdit
18116: 366 unit uPSI_devcutils;                    //DevC
18117: 367 unit uPSI_SynEditMiscClasses;           //SynEdit
18118: 368 unit uPSI_SynEditRegexSearch;           //SynEdit
18119: 369 unit uPSI_SynEditHighlighter;           //SynEdit
18120: 370 unit uPSI_SynHighlighterPas;            //SynEdit
18121: 371 unit uPSI_JvSearchFiles;                //JCL
18122: 372 unit uPSI_SynHighlighterAny;            //Lazarus
18123: 373 unit uPSI_SynEditKeyCmds;               //SynEdit
18124: 374 unit uPSI_SynEditMiscProcs,             //SynEdit
18125: 375 unit uPSI_SynEditKbdHandler             //SynEdit
18126: 376 unit uPSI_JvAppInst,                    //JCL
18127: 377 unit uPSI_JvAppEvent;                   //JCL
18128: 378 unit uPSI_JvAppCommand;                 //JCL
18129: 379 unit uPSI_JvAnimTitle;                  //JCL
18130: 380 unit uPSI_JvAnimatedImage;              //JCL
18131: 381 unit uPSI_SynEditExport;                //SynEdit
18132: 382 unit uPSI_SynExportHTML;                //SynEdit
18133: 383 unit uPSI_SynExportRTF;                 //SynEdit
18134: 384 unit uPSI_SynEditSearch;                //SynEdit
18135: 385 unit uPSI_fMain_back                    //maXbox;
18136: 386 unit uPSI_JvZoom;                       //JCL
18137: 387 unit uPSI_PMrand;                       //PM
18138: 388 unit uPSI_JvSticker;                    //JCL
18139: 389 unit uPSI_XmlVerySimple;                //mX4
18140: 390 unit uPSI_Services;                     //ExtPascal
18141: 391 unit uPSI_ExtPascalUtils;               //ExtPascal
18142: 392 unit uPSI_SocketsDelphi;                //ExtPascal
18143: 393 unit uPSI_StBarC;                       //SysTools
18144: 394 unit uPSI_StDbBarC;                     //SysTools
18145: 395 unit uPSI_StBarPN;                      //SysTools
18146: 396 unit uPSI_StDbPNBC;                     //SysTools
18147: 397 unit uPSI_StDb2DBC;                     //SysTools
18148: 398 unit uPSI_StMoney;                      //SysTools
18149: 399 unit uPSI_JvForth;                      //JCL
18150: 400 unit uPSI_RestRequest;                  //mX4
18151: 401 unit uPSI_HttpRESTConnectionIndy;       //mX4
18152: 402 unit uPSI_JvXmlDatabase; //update       //JCL
18153: 403 unit uPSI_StAstro;                      //SysTools
18154: 404 unit uPSI_StSort;                       //SysTools
18155: 405 unit uPSI_StDate;                       //SysTools
18156: 406 unit uPSI_StDateSt;                     //SysTools
18157: 407 unit uPSI_StBase;                       //SysTools
18158: 408 unit uPSI_StVInfo;                      //SysTools
18159: 409 unit uPSI_JvBrowseFolder;               //JCL
18160: 410 unit uPSI_JvBoxProcs;                   //JCL
18161: 411 unit uPSI_urandom; (unit uranuvag;)     //DMath
18162: 412 unit uPSI_usimann; (unit ugenalg;)      //DMath
18163: 413 unit uPSI_JvHighlighter;                //JCL
18164: 414 unit uPSI_Diff;                         //mX4
18165: 415 unit uPSI_SpringWinAPI;                 //DSpring
18166: 416 unit uPSI_StBits;                       //SysTools
18167: 417 unit uPSI_TomDBQue;                     //mX4
18168: 418 unit uPSI_MultilangTranslator;          //mX4
18169: 419 unit uPSI_HyperLabel;                   //mX4
18170: 420 unit uPSI_Starter;                      //mX4
18171: 421 unit uPSI_FileAssocs;                   //devC
18172: 422 unit uPSI_devFileMonitorX;              //devC
18173: 423 unit uPSI_devrun;                       //devC
18174: 424 unit uPSI_devExec;                      //devC
18175: 425 unit uPSI_oysUtils;                     //devC
18176: 426 unit uPSI_DosCommand;                   //devC
18177: 427 unit uPSI_CppTokenizer;                 //devC
18178: 428 unit uPSI_JvHLParser;                   //devC
18179: 429 unit uPSI_JclMapi;                      //JCL
18180: 430 unit uPSI_JclShell;                     //JCL
18181: 431 unit uPSI_JclCOM;                       //JCL
18182: 432 unit uPSI_GR32_Math;                    //Graphics32
```

```
18183: 433 unit uPSI_GR32_LowLevel;                 //Graphics32
18184: 434 unit uPSI_SimpleHl;                      //mX4
18185: 435 unit uPSI_GR32_Filters,                  //Graphics32
18186: 436 unit uPSI_GR32_VectorMaps;               //Graphics32
18187: 437 unit uPSI_cXMLFunctions;                 //Fundamentals 4
18188: 438 unit uPSI_JvTimer;                       //JCL
18189: 439 unit uPSI_cHTTPUtils;                    //Fundamentals 4
18190: 440 unit uPSI_cTLSUtils;                     //Fundamentals 4
18191: 441 unit uPSI_JclGraphics;                   //JCL
18192: 442 unit uPSI_JclSynch;                      //JCL
18193: 443 unit uPSI_IdTelnet;                      //Indy
18194: 444 unit uPSI_IdTelnetServer,                //Indy
18195: 445 unit uPSI_IdEcho,                        //Indy
18196: 446 unit uPSI_IdEchoServer,                  //Indy
18197: 447 unit uPSI_IdEchoUDP,                     //Indy
18198: 448 unit uPSI_IdEchoUDPServer,               //Indy
18199: 449 unit uPSI_IdSocks,                       //Indy
18200: 450 unit uPSI_IdAntiFreezeBase;              //Indy
18201: 451 unit uPSI_IdHostnameServer;              //Indy
18202: 452 unit uPSI_IdTunnelCommon,                //Indy
18203: 453 unit uPSI_IdTunnelMaster,                //Indy
18204: 454 unit uPSI_IdTunnelSlave,                 //Indy
18205: 455 unit uPSI_IdRSH,                         //Indy
18206: 456 unit uPSI_IdRSHServer,                   //Indy
18207: 457 unit uPSI_Spring_Cryptography_Utils;     //Spring4Delphi
18208: 458 unit uPSI_MapReader,                     //devC
18209: 459 unit uPSI_LibTar,                        //devC
18210: 460 unit uPSI_IdStack;                       //Indy
18211: 461 unit uPSI_IdBlockCipherIntercept;        //Indy
18212: 462 unit uPSI_IdChargenServer;               //Indy
18213: 463 unit uPSI_IdFTPServer,                   //Indy
18214: 464 unit uPSI_IdException,                   //Indy
18215: 465 unit uPSI_utexplot;                      //DMath
18216: 466 unit uPSI_uwinstr;                       //DMath
18217: 467 unit uPSI_VarRecUtils;                   //devC
18218: 468 unit uPSI_JvStringListToHtml,            //JCL
18219: 469 unit uPSI_JvStringHolder,                //JCL
18220: 470 unit uPSI_IdCoder;                       //Indy
18221: 471 unit uPSI_SynHighlighterDfm;             //Synedit
18222: 472 unit uHighlighterProcs; in 471           //Synedit
18223: 473 unit uPSI_LazFileUtils,                  //LCL
18224: 474 unit uPSI_IDECmdLine;                    //LCL
18225: 475 unit uPSI_lazMasks;                      //LCL
18226: 476 unit uPSI_ip_misc;                       //mX4
18227: 477 unit uPSI_Barcode;                       //LCL
18228: 478 unit uPSI_SimpleXML;                     //LCL
18229: 479 unit uPSI_JclIniFiles;                   //JCL
18230: 480 unit uPSI_D2XXUnit;  {$X-}               //FTDI
18231: 481 unit uPSI_JclDateTime;                   //JCL
18232: 482 unit uPSI_JclEDI;                        //JCL
18233: 483 unit uPSI_JclMiscel2;                    //JCL
18234: 484 unit uPSI_JclValidation;                 //JCL
18235: 485 unit uPSI_JclAnsiStrings; {-PString}     //JCL
18236: 486 unit uPSI_SynEditMiscProcs2;             //Synedit
18237: 487 unit uPSI_JclStreams;                    //JCL
18238: 488 unit uPSI_QRCode;                        //mX4
18239: 489 unit uPSI_BlockSocket;                   //ExtPascal
18240: 490 unit uPSI_Masks,Utils                    //VCL
18241: 491 unit uPSI_synautil;                      //Synapse!
18242: 492 unit uPSI_JclMath_Class;                 //JCL RTL
18243: 493 unit ugamdist; //Gamma function         //DMath
18244: 494 unit uibeta, ucorrel; //IBeta           //DMath
18245: 495 unit uPSI_SRMgr;                         //mX4
18246: 496 unit uPSI_HotLog;                        //mX4
18247: 497 unit uPSI_DebugBox;                      //mX4
18248: 498 unit uPSI_ustrings;                      //DMath
18249: 499 unit uPSI_uregtest;                      //DMath
18250: 500 unit uPSI_usimplex;                      //DMath
18251: 501 unit uPSI_uhyper;                        //DMath
18252: 502 unit uPSI_IdHL7;                         //Indy
18253: 503 unit uPSI_IdIPMCastBase,                 //Indy
18254: 504 unit uPSI_IdIPMCastServer;               //Indy
18255: 505 unit uPSI_IdIPMCastClient;               //Indy
18256: 506 unit uPSI_unlfit; //nlregression        //DMath
18257: 507 unit uPSI_IdRawHeaders;                  //Indy
18258: 508 unit uPSI_IdRawClient;                   //Indy
18259: 509 unit uPSI_IdRawFunctions;                //Indy
18260: 510 unit uPSI_IdTCPStream;                   //Indy
18261: 511 unit uPSI_IdSNPP;                        //Indy
18262: 512 unit uPSI_St2DBarC;                      //SysTools
18263: 513 unit uPSI_ImageWin;  //FTL              //VCL
18264: 514 unit uPSI_CustomDrawTreeView; //FTL     //VCL
18265: 515 unit uPSI_GraphWin;  //FTL              //VCL
18266: 516 unit uPSI_actionMain;  //FTL            //VCL
18267: 517 unit uPSI_StSpawn;                       //SysTools
18268: 518 unit uPSI_CtlPanel;                      //VCL
18269: 519 unit uPSI_IdLPR;                         //Indy
18270: 520 unit uPSI_SockRequestInterpreter;        //Indy
18271: 521 unit uPSI_ulambert;                      //DMath
```

```
18272: 522 unit uPSI_ucholesk;                    //DMath
18273: 523 unit uPSI_SimpleDS;                    //VCL
18274: 524 unit uPSI_DBXSqlScanner;               //VCL
18275: 525 unit uPSI_DBXMetaDataUtil;             //VCL
18276: 526 unit uPSI_Chart;                       //TEE
18277: 527 unit uPSI_TeeProcs;                    //TEE
18278: 528 unit mXBDEUtils;                       //mX4
18279: 529 unit uPSI_MDIEdit;                     //VCL
18280: 530 unit uPSI_CopyPrsr;                    //VCL
18281: 531 unit uPSI_SockApp;                     //VCL
18282: 532 unit uPSI_AppEvnts;                    //VCL
18283: 533 unit uPSI_ExtActns;                    //VCL
18284: 534 unit uPSI_TeEngine;                    //TEE
18285: 535 unit uPSI_CoolMain; //browser          //VCL
18286: 536 unit uPSI_StCRC;                       //SysTools
18287: 537 unit uPSI_StDecMth2;                   //SysTools
18288: 538 unit uPSI_frmExportMain;               //Synedit
18289: 539 unit uPSI_SynDBEdit;                   //Synedit
18290: 540 unit uPSI_SynEditWildcardSearch;       //Synedit
18291: 541 unit uPSI_BoldComUtils;                //BOLD
18292: 542 unit uPSI_BoldIsoDateTime;             //BOLD
18293: 543 unit uPSI_BoldGUIDUtils; //inCOMUtils  //BOLD
18294: 544 unit uPSI_BoldXMLRequests;             //BOLD
18295: 545 unit uPSI_BoldStringList;              //BOLD
18296: 546 unit uPSI_BoldFileHandler;             //BOLD
18297: 547 unit uPSI_BoldContainers;              //BOLD
18298: 548 unit uPSI_BoldQueryUserDlg;            //BOLD
18299: 549 unit uPSI_BoldWinINet;                 //BOLD
18300: 550 unit uPSI_BoldQueue;                   //BOLD
18301: 551 unit uPSI_JvPcx;                       //JCL
18302: 552 unit uPSI_IdWhois;                     //Indy
18303: 553 unit uPSI_IdWhoIsServer;               //Indy
18304: 554 unit uPSI_IdGopher;                    //Indy
18305: 555 unit uPSI_IdDateTimeStamp;             //Indy
18306: 556 unit uPSI_IdDayTimeServer;             //Indy
18307: 557 unit uPSI_IdDayTimeUDP;                //Indy
18308: 558 unit uPSI_IdDayTimeUDPServer;          //Indy
18309: 559 unit uPSI_IdDICTServer;                //Indy
18310: 560 unit uPSI_IdDiscardServer;             //Indy
18311: 561 unit uPSI_IdDiscardUDPServer;          //Indy
18312: 562 unit uPSI_IdMappedFTP;                 //Indy
18313: 563 unit uPSI_IdMappedPortTCP;             //Indy
18314: 564 unit uPSI_IdGopherServer;              //Indy
18315: 565 unit uPSI_IdQotdServer;                //Indy
18316: 566 unit uPSI_JvRgbToHtml;                 //JCL
18317: 567 unit uPSI_JvRemLog,                    //JCL
18318: 568 unit uPSI_JvSysComp;                   //JCL
18319: 569 unit uPSI_JvTMTL;                      //JCL
18320: 570 unit uPSI_JvWinampAPI;                 //JCL
18321: 571 unit uPSI_MSysUtils;                   //mX4
18322: 572 unit uPSI_ESBMaths;                    //ESB
18323: 573 unit uPSI_ESBMaths2;                   //ESB
18324: 574 unit uPSI_uLkJSON;                     //Lk
18325: 575 unit uPSI_ZURL;  //Zeos                //Zeos
18326: 576 unit uPSI_ZSysUtils;                   //Zeos
18327: 577 unit unaUtils internals                //UNA
18328: 578 unit uPSI_ZMatchPattern;               //Zeos
18329: 579 unit uPSI_ZClasses;                    //Zeos
18330: 580 unit uPSI_ZCollections;                //Zeos
18331: 581 unit uPSI_ZEncoding;                   //Zeos
18332: 582 unit uPSI_IdRawBase;                   //Indy
18333: 583 unit uPSI_IdNTLM;                      //Indy
18334: 584 unit uPSI_IdNNTP;                      //Indy
18335: 585 unit uPSI_usniffer; //PortScanForm     //mX4
18336: 586 unit uPSI_IdCoderMIME;                 //Indy
18337: 587 unit uPSI_IdCoderUUE;                  //Indy
18338: 588 unit uPSI_IdCoderXXE;                  //Indy
18339: 589 unit uPSI_IdCoder3to4;                 //Indy
18340: 590 unit uPSI_IdCookie;                    //Indy
18341: 591 unit uPSI_IdCookieManager;             //Indy
18342: 592 unit uPSI_WDosSocketUtils;             //WDos
18343: 593 unit uPSI_WDosPlcUtils;                //WDos
18344: 594 unit uPSI_WDosPorts;                   //WDos
18345: 595 unit uPSI_WDosResolvers;               //WDos
18346: 596 unit uPSI_WDosTimers;                  //WDos
18347: 597 unit uPSI_WDosPlcs;                    //WDos
18348: 598 unit uPSI_WDosPneumatics;              //WDos
18349: 599 unit uPSI_IdFingerServer;              //Indy
18350: 600 unit uPSI_IdDNSResolver;               //Indy
18351: 601 unit uPSI_IdHTTPWebBrokerBridge;       //Indy
18352: 602 unit uPSI_IdIntercept;                 //Indy
18353: 603 unit uPSI_IdIPMCastBase;               //Indy
18354: 604 unit uPSI_IdLogBase;                   //Indy
18355: 605 unit uPSI_IdIOHandlerStream;           //Indy
18356: 606 unit uPSI_IdMappedPortUDP;             //Indy
18357: 607 unit uPSI_IdQOTDUDPServer;             //Indy
18358: 608 unit uPSI_IdQOTDUDP;                   //Indy
18359: 609 unit uPSI_IdSysLog;                    //Indy
18360: 610 unit uPSI_IdSysLogServer;              //Indy
```

```
18361: 611 unit uPSI_IdSysLogMessage;                //Indy
18362: 612 unit uPSI_IdTimeServer;                   //Indy
18363: 613 unit uPSI_IdTimeUDP;                      //Indy
18364: 614 unit uPSI_IdTimeUDPServer;                //Indy
18365: 615 unit uPSI_IdUserAccounts;                 //Indy
18366: 616 unit uPSI_TextUtils;                      //mX4
18367: 617 unit uPSI_MandelbrotEngine;               //mX4
18368: 618 unit uPSI_delphi_arduino_Unit1;           //mX4
18369: 619 unit uPSI_DTDSchema2;                     //mX4
18370: 620 unit uPSI_fplotMain;                      //DMath
18371: 621 unit uPSI_FindFileIter;                   //mX4
18372: 622 unit uPSI_PppState;  (JclStrHashMap)      //PPP
18373: 623 unit uPSI_PppParser;                      //PPP
18374: 624 unit uPSI_PppLexer;                       //PPP
18375: 625 unit uPSI_PCharUtils;                     //PPP
18376: 626 unit uPSI_uJSON;                          //WU
18377: 627 unit uPSI_JclStrHashMap;                  //JCL
18378: 628 unit uPSI_JclHookExcept;                  //JCL
18379: 629 unit uPSI_EncdDecd;                       //VCL
18380: 630 unit uPSI_SockAppReg;                     //VCL
18381: 631 unit uPSI_PJFileHandle;                   //PJ
18382: 632 unit uPSI_PJEnvVars;                      //PJ
18383: 633 unit uPSI_PJPipe;                         //PJ
18384: 634 unit uPSI_PJPipeFilters;                  //PJ
18385: 635 unit uPSI_PJConsoleApp;                   //PJ
18386: 636 unit uPSI_UConsoleAppEx;                  //PJ
18387: 637 unit uPSI_DbxSocketChannelNative,         //VCL
18388: 638 unit uPSI_DbxDataGenerator,               //VCL
18389: 639 unit uPSI_DBXClient;                      //VCL
18390: 640 unit uPSI_IdLogEvent;                     //Indy
18391: 641 unit uPSI_Reversi;                        //mX4
18392: 642 unit uPSI_Geometry;                       //mX4
18393: 643 unit uPSI_IdSMTPServer;                   //Indy
18394: 644 unit uPSI_Textures;                       //mX4
18395: 645 unit uPSI_IBX;                            //VCL
18396: 646 unit uPSI_IWDBCommon;                     //VCL
18397: 647 unit uPSI_SortGrid;                       //mX4
18398: 648 unit uPSI_IB;                             //VCL
18399: 649 unit uPSI_IBScript;                       //VCL
18400: 650 unit uPSI_JvCSVBaseControls;              //JCL
18401: 651 unit uPSI_Jvg3DColors;                    //JCL
18402: 652 unit uPSI_JvHLEditor;  //beat             //JCL
18403: 653 unit uPSI_JvShellHook;                    //JCL
18404: 654 unit uPSI_DBCommon2                       //VCL
18405: 655 unit uPSI_JvSHFileOperation;              //JCL
18406: 656 unit uPSI_uFilexport;                     //mX4
18407: 657 unit uPSI_JvDialogs;                      //JCL
18408: 658 unit uPSI_JvDBTreeView;                   //JCL
18409: 659 unit uPSI_JvDBUltimGrid;                  //JCL
18410: 660 unit uPSI_JvDBQueryParamsForm;            //JCL
18411: 661 unit uPSI_JvExControls;                   //JCL
18412: 662 unit uPSI_JvBDEMemTable;                  //JCL
18413: 663 unit uPSI_JvCommStatus;                   //JCL
18414: 664 unit uPSI_JvMailSlots2;                   //JCL
18415: 665 unit uPSI_JvgWinMask;                     //JCL
18416: 666 unit uPSI_StEclpse;                       //SysTools
18417: 667 unit uPSI_StMime;                         //SysTools
18418: 668 unit uPSI_StList;                         //SysTools
18419: 669 unit uPSI_StMerge;                        //SysTools
18420: 670 unit uPSI_StStrS;                         //SysTools
18421: 671 unit uPSI_StTree,                         //SysTools
18422: 672 unit uPSI_StVArr;                         //SysTools
18423: 673 unit uPSI_StRegIni;                       //SysTools
18424: 674 unit uPSI_urkf;                           //DMath
18425: 675 unit uPSI_usvd;                           //DMath
18426: 676 unit uPSI_DepWalkUtils;                   //JCL
18427: 677 unit uPSI_OptionsFrm;                     //JCL
18428: 678 unit yuvconverts;                         //mX4
18429: 679 uPSI_JvPropAutoSave;                      //JCL
18430: 680 uPSI_AclAPI;                              //alcinoe
18431: 681 uPSI_AviCap;                              //alcinoe
18432: 682 uPSI_ALAVLBinaryTree;                     //alcinoe
18433: 683 uPSI_ALFcnMisc;                           //alcinoe
18434: 684 uPSI_ALStringList;                        //alcinoe
18435: 685 uPSI_ALQuickSortList;                     //alcinoe
18436: 686 uPSI_ALStaticText;                        //alcinoe
18437: 687 uPSI_ALJSONDoc;                           //alcinoe
18438: 688 uPSI_ALGSMComm;                           //alcinoe
18439: 689 uPSI_ALWindows;                           //alcinoe
18440: 690 uPSI_ALMultiPartFormDataParser;           //alcinoe
18441: 691 uPSI_ALHttpCommon;                        //alcinoe
18442: 692 uPSI_ALWebSpider,                         //alcinoe
18443: 693 uPSI_ALHttpClient;                        //alcinoe
18444: 694 uPSI_ALFcnHTML;                           //alcinoe
18445: 695 uPSI_ALFTPClient;                         //alcinoe
18446: 696 uPSI_ALInternetMessageCommon;             //alcinoe
18447: 697 uPSI_ALWininetHttpClient;                 //alcinoe
18448: 698 uPSI_ALWinInetFTPClient;                  //alcinoe
18449: 699 uPSI_ALWinHttpWrapper;                    //alcinoe
```

```
18450: 700 uPSI_ALWinHttpClient;                    //alcinoe
18451: 701 uPSI_ALFcnWinSock;                       //alcinoe
18452: 702 uPSI_ALFcnSQL;                          //alcinoe
18453: 703 uPSI_ALFcnCGI;                          //alcinoe
18454: 704 uPSI_ALFcnExecute;                      //alcinoe
18455: 705 uPSI_ALFcnFile;                         //alcinoe
18456: 706 uPSI_ALFcnMime;                         //alcinoe
18457: 707 uPSI_ALPhpRunner;                       //alcinoe
18458: 708 uPSI_ALGraphic;                         //alcinoe
18459: 709 uPSI_ALIniFiles;                        //alcinoe
18460: 710 uPSI_ALMemCachedClient;                 //alcinoe
18461: 711 unit uPSI_MyGrids;                      //mX4
18462: 712 uPSI_ALMultiPartMixedParser             //alcinoe
18463: 713 uPSI_ALSMTPClient                       //alcinoe
18464: 714 uPSI_ALNNTPClient;                      //alcinoe
18465: 715 uPSI_ALHintBalloon;                     //alcinoe
18466: 716 unit uPSI_ALXmlDoc;                     //alcinoe
18467: 717 unit uPSI_IPCThrd;                      //VCL
18468: 718 unit uPSI_MonForm;                      //VCL
18469: 719 unit uPSI_TeCanvas;                     //Orpheus
18470: 720 unit uPSI_Ovcmisc;                      //Orpheus
18471: 721 unit uPSI_ovcfiler;                     //Orpheus
18472: 722 unit uPSI_ovcstate;                     //Orpheus
18473: 723 unit uPSI_ovccoco;                      //Orpheus
18474: 724 unit uPSI_ovcrvexp;                     //Orpheus
18475: 725 unit uPSI_OvcFormatSettings;            //Orpheus
18476: 726 unit uPSI_OvcUtils;                     //Orpheus
18477: 727 unit uPSI_ovcstore;                     //Orpheus
18478: 728 unit uPSI_ovcstr;                       //Orpheus
18479: 729 unit uPSI_ovcmru;                       //Orpheus
18480: 730 unit uPSI_ovccmd;                       //Orpheus
18481: 731 unit uPSI_ovctimer;                     //Orpheus
18482: 732 unit uPSI_ovcintl;                      //Orpheus
18483: 733 uPSI_AfCircularBuffer;                  //AsyncFree
18484: 734 uPSI_AfUtils;                           //AsyncFree
18485: 735 uPSI_AfSafeSync;                        //AsyncFree
18486: 736 uPSI_AfComPortCore;                     //AsyncFree
18487: 737 uPSI_AfComPort;                         //AsyncFree
18488: 738 uPSI_AfPortControls;                    //AsyncFree
18489: 739 uPSI_AfDataDispatcher;                  //AsyncFree
18490: 740 uPSI_AfViewers;                         //AsyncFree
18491: 741 uPSI_AfDataTerminal;                    //AsyncFree
18492: 742 uPSI_SimplePortMain;                    //AsyncFree
18493: 743 unit uPSI_ovcclock;                     //Orpheus
18494: 744 unit uPSI_o32intlst;                    //Orpheus
18495: 745 unit uPSI_o32ledlabel;                  //Orpheus
18496: 746 unit uPSI_AlMySqlClient;                //alcinoe
18497: 747 unit uPSI_ALFBXClient;                  //alcinoe
18498: 748 unit uPSI_ALFcnSQL;                     //alcinoe
18499: 749 unit uPSI_AsyncTimer;                   //mX4
18500: 750 unit uPSI_ApplicationFileIO;            //mX4
18501: 751 unit uPSI_PsAPI;                        //VCLé
18502: 752 uPSI_ovcuser;                           //Orpheus
18503: 753 uPSI_ovcurl;                            //Orpheus
18504: 754 uPSI_ovcvlb;                            //Orpheus
18505: 755 uPSI_ovccolor;                          //Orpheus
18506: 756 uPSI_ALFBXLib,                          //alcinoe
18507: 757 uPSI_ovcmeter;                          //Orpheus
18508: 758 uPSI_ovcpeakm;                          //Orpheus
18509: 759 uPSI_O32BGSty;                          //Orpheus
18510: 760 uPSI_ovcBidi;                           //Orpheus
18511: 761 uPSI_ovctcary;                          //Orpheus
18512: 762 uPSI_DXPUtils;                          //mX4
18513: 763 uPSI_ALMultiPartBaseParser;             //alcinoe
18514: 764 uPSI_ALMultiPartAlternativeParser;      //alcinoe
18515: 765 uPSI_ALPOP3Client;                      //alcinoe
18516: 766 uPSI_SmallUtils;                        //mX4
18517: 767 uPSI_MakeApp;                           //mX4
18518: 768 uPSI_O32MouseMon;                       //Orpheus
18519: 769 uPSI_OvcCache;                          //Orpheus
18520: 770 uPSI_ovccalc;                           //Orpheus
18521: 771 uPSI_Joystick,                          //OpenGL
18522: 772 uPSI_ScreenSaver;                       //OpenGL
18523: 773 uPSI_XCollection,                       //OpenGL
18524: 774 uPSI_Polynomials,                       //OpenGL
18525: 775 uPSI_PersistentClasses, //9.86         //OpenGL
18526: 776 uPSI_VectorLists;                       //OpenGL
18527: 777 uPSI_XOpenGL,                          //OpenGL
18528: 778 uPSI_MeshUtils;                         //OpenGL
18529: 779 unit uPSI_JclSysUtils;                  //JCL
18530: 780 unit uPSI_JclBorlandTools;              //JCL
18531: 781 unit JclFileUtils_max;                  //JCL
18532: 782 uPSI_AfDataControls,                    //AsyncFree
18533: 783 uPSI_GLSilhouette;                      //OpenGL
18534: 784 uPSI_JclSysUtils_class;                 //JCL
18535: 785 uPSI_JclFileUtils_class;                //JCL
18536: 786 uPSI_FileUtil;                          //JCL
18537: 787 uPSI_changefind;                        //mX4
18538: 788 uPSI_cmdIntf;                           //mX4
```

```
18539: 789 uPSI_fservice;                          //mX4
18540: 790 uPSI_Keyboard;                          //OpenGL
18541: 791 uPSI_VRMLParser,                        //OpenGL
18542: 792 uPSI_GLFileVRML,                       //OpenGL
18543: 793 uPSI_Octree;                           //OpenGL
18544: 794 uPSI_GLPolyhedron,                     //OpenGL
18545: 795 uPSI_GLCrossPlatform;                 //OpenGL
18546: 796 uPSI_GLParticles;                     //OpenGL
18547: 797 uPSI_GLNavigator;                     //OpenGL
18548: 798 uPSI_GLStarRecord;                    //OpenGL
18549: 799 uPSI_GLTextureCombiners;              //OpenGL
18550: 800 uPSI_GLCanvas;                        //OpenGL
18551: 801 uPSI_GeometryBB;                      //OpenGL
18552: 802 uPSI_GeometryCoordinates;             //OpenGL
18553: 803 uPSI_VectorGeometry;                  //OpenGL
18554: 804 uPSI_BumpMapping;                     //OpenGL
18555: 805 uPSI_TGA;                             //OpenGL
18556: 806 uPSI_GLVectorFileObjects;             //OpenGL
18557: 807 uPSI_IMM;                             //VCL
18558: 808 uPSI_CategoryButtons;                 //VCL
18559: 809 uPSI_ButtonGroup;                     //VCL
18560: 810 uPSI_DbExcept;                        //VCL
18561: 811 uPSI_AxCtrls;                         //VCL
18562: 812 uPSI_GL_actorUnit1;                   //OpenGL
18563: 813 uPSI_StdVCL;                          //VCL
18564: 814 unit CurvesAndSurfaces;               //OpenGL
18565: 815 uPSI_DataAwareMain;                   //AsyncFree
18566:
18567:
18568:
18569: ////////////////////////////////////////////////////////////////////////////
18570: //Form Template Library FTL
18571: ////////////////////////////////////////////////////////////////////////////
18572:
18573: 25 FTL For Form Building out of the Script, eg. 399_form_templates.txt
18574:
18575: 045 unit uPSI_VListView                     TFormListView;
18576: 263 unit uPSI_JvProfiler32;                 TProfReport
18577: 270 unit uPSI_ImgList;                      TCustomImageList
18578: 278 unit uPSI_JvImageWindow;                TJvImageWindow
18579: 317 unit uPSI_JvParserForm;                 TJvHTMLParserForm
18580: 497 unit uPSI_DebugBox;                     TDebugBox
18581: 513 unit uPSI_ImageWin;                     TImageForm, TImageForm2
18582: 514 unit uPSI_CustomDrawTreeView;           TCustomDrawForm
18583: 515 unit uPSI_GraphWin;                     TGraphWinForm
18584: 516 unit uPSI_actionMain;                   TActionForm
18585: 518 unit uPSI_CtlPanel;                     TAppletApplication
18586: 529 unit uPSI_MDIEdit;                      TEditForm
18587: 535 unit uPSI_CoolMain; {browser}          TWebMainForm
18588: 538 unit uPSI_frmExportMain;               TSynexportForm
18589: 585 unit uPSI_usniffer; {//PortScanForm}   TSniffForm
18590: 600 unit uPSI_ThreadForm;                  TThreadSortForm;
18591: 618 unit uPSI_delphi_arduino_Unit1;        TLEDForm
18592: 620 unit uPSI_fplotMain;                   TfplotForm1
18593: 660 unit uPSI_JvDBQueryParamsForm;         TJvQueryParamsDialog
18594: 677 unit uPSI_OptionsFrm;                  TfrmOptions;
18595: 718 unit uPSI_MonForm;                     TMonitorForm
18596: 742 unit uPSI_SimplePortMain;             TPortForm1
18597: 770 unit uPSI_ovccalc;                     TOvcCalculator  //widget
18598: 810 unit uPSI_DbExcept;                    TDbEngineErrorDlg
18599:
18600:
18601: ex.:with TEditForm.create(self) do begin
18602:       caption:= 'Template Form Tester';
18603:       FormStyle:= fsStayOnTop;
18604:       with editor do begin
18605:         Lines.LoadFromFile(Exepath+'\docs\Readme_rus_mX2.rtf
18606:         SelStart:= 0;
18607:         Modified:= False;
18608:       end;
18609:     end;
18610:   with TWebMainForm.create(self) do begin
18611:     URLs.Text:= 'http://www.kleiner.ch';
18612:     URLsClick(self); Show;
18613:   end;
18614:   with TSynexportForm.create(self) do begin
18615:     Caption:= 'Synexport HTML RTF tester';
18616:     Show;
18617:   end;
18618:   with TThreadSortForm.create(self) do begin
18619:      showmodal; free;
18620:   end;
18621:
18622:   with TCustomDrawForm.create(self) do begin
18623:        width:=820; height:=820;
18624:        image1.height:= 600; //add properties
18625:        image1.picture.bitmap:= image2.picture.bitmap;
18626:        //SelectionBackground1Click(self) CustomDraw1Click(self);
18627:        Background1.click;
```

```
18628:        bitmap1.click;
18629:        Tile1.click;
18630:        Showmodal;
18631:        Free;
18632:      end;
18633:
18634:    with TfplotForm1.Create(self) do begin
18635:      BtnPlotClick(self);
18636:      Showmodal; Free;
18637:    end;
18638:
18639:  with TOvcCalculator.create(self) do begin
18640:     parent:= aForm;
18641:   //free;
18642:     setbounds(550,510,200,150);
18643:     displaystr:= 'maXcalc';
18644:   end;
18645:
18646:
18647: ////////////////////////////////////////////////////////////////////////////
18648: All maXbox Tutorials Table of Content 2014
18649: ////////////////////////////////////////////////////////////////////////////
18650:     Tutorial 00 Function-Coding  (Blix the Programmer)
18651:     Tutorial 01 Procedural-Coding
18652:     Tutorial 02 OO-Programming
18653:     Tutorial 03 Modular Coding
18654:     Tutorial 04 UML Use Case Coding
18655:     Tutorial 05 Internet Coding
18656:     Tutorial 06 Network Coding
18657:     Tutorial 07 Game Graphics Coding
18658:     Tutorial 08 Operating System Coding
18659:     Tutorial 09 Database Coding
18660:     Tutorial 10 Statistic Coding
18661:     Tutorial 11 Forms Coding
18662:     Tutorial 12 SQL DB Coding
18663:     Tutorial 13 Crypto Coding
18664:     Tutorial 14 Parallel Coding
18665:     Tutorial 15 Serial RS232 Coding
18666:     Tutorial 16 Event Driven Coding
18667:     Tutorial 17 Web Server Coding
18668:     Tutorial 18 Arduino System Coding
18669:     Tutorial 18_3 RGB LED System Coding
18670:     Tutorial 19 WinCOM /Arduino Coding
18671:     Tutorial 20 Regular Expressions RegEx
18672:     Tutorial 21 Android Coding (coming 2013)
18673:     Tutorial 22 Services Programming
18674:     Tutorial 23 Real Time Systems
18675:     Tutorial 24 Clean Code
18676:     Tutorial 25 maXbox Configuration I+II
18677:     Tutorial 26 Socket Programming with TCP
18678:     Tutorial 27 XML & TreeView
18679:     Tutorial 28 DLL Coding (coming 2014)
18680:     Tutorial 29 UML Scripting (coming 2014)
18681:     Tutorial 30 Web of Things (coming 2014)
18682:     Tutorial 31 Closures (coming 2014)
18683:     Tutorial 32 SQL Firebird (coming 2014)
18684:
18685:
18686: ref Docu for all Type Class and Const in maXbox_types.pdf
18687: using Docu for this file is maxbox_functions_all.pdf
18688: PEP - Pascal Education Program  Lib Lab
18689:
18690:
18691: http://stackoverflow.com/tags/pascalscript/hot
18692: http://www.jrsoftware.org/ishelp/index.php?topic=scriptfunctions
18693: http://sourceforge.net/projects/maXbox   #locs:15162
18694: http://sourceforge.net/apps/mediawiki/maXbox
18695: http://www.blaisepascal.eu/
18696: https://github.com/maxkleiner/maXbox3.git
18697: http://www.heise.de/download/maxbox-1176464.html
18698: http://www.softpedia.com/get/Programming/Other-Programming-Files/maXbox.shtml
18699:
18700:   ---- bigbitbox code_cleared_checked----
```