

```

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: 19483136 V3.9.9.91 March 2014 To EKON/BASTA/JAX 18 2014
10: *****Now the Funclist*****
11: Funclist Function : 11818 //10766 //10165 (7648)
12: *****Now the Proclist*****
13: Proclist Procedure Size is: 7421 //6792 //6401 //4752 (4552)
14: *****Now Constructors*****
15: Constructlist Constructor Size is: 1219 //995 //777 (689 /513 /494)
16: def head:max: maXbox7: 10.03.2014 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: 20458! Constructor, Function and Procedure
21: AExtraxt of EXE Functions of maxbox3.exe, locs of file = 19301
22: ASize of EXE: 19483136 (16586240) (13511680) (13023744)
23: SHA1 Hash of maXbox 3.9.9.91: 8099E25F2508B262E909B76EDF7BB301AFFA1864
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 : TComboBoxItem
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 : TINDEDEF
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 AddParamsSQLForDetail(Params:TParams;SQL:WideStr;Native:Bool;QuoteChar:WideString):WideString
105: Function Addr64ToAddr32(const Value: TJclAddr64): TJclAddr32;
106: Function Addr32ToAddr64(const Value: TJclAddr32): TJclAddr64;
107: function AdjustLineBreaksS(const S: string): string
108: TTextLineBreakStyle', '(tlbsLF, tlbsCRLF)
109: Function AdjustLineBreaks(const S: string; Style: TTextLineBreakStyle): string;
110: Function AllData : string
111: function AllocMemCount: integer;
112: function AllocMemSize: integer;
113: Function AllocPatternBitmap( BkColor, FgColor : TColor) : TBitmap
114: Function AllowRegKeyForEveryone( Key : HKEY; Path : string) : Boolean
115: Function AlphaComponent( const Color32 : TColor32) : Integer
116: Function AlphaSort : Boolean
117: Function AlphaSort( ARecurse : Boolean) : Boolean
118: Function AnsiCat( const x, y : AnsiString) : AnsiString
119: Function AnsiCompareFileName( S1, S2 : string) : Integer
120: function AnsiCompareFileName(const S1: string; const S2: string): Integer)
121: Function AnsiCompareStr( S1, S2 : string) : Integer
122: function AnsiCompareStr(const S1: string; const S2: string): Integer;)
123: Function AnsiCompareText( S1, S2 : string) : Integer
124: function AnsiCompareText(const S1: string; const S2: string): Integer;)
125: Function AnsiContainsStr( const AText, ASubText : string) : Boolean
126: Function AnsiContainsText( const AText, ASubText : string) : Boolean
127: Function AnsiCopy( const src : AnsiString; index, count : Integer) : AnsiString
128: Function AnsiDequotedStr( S : string; AQuote : Char) : string
129: Function AnsiEndsStr( const ASubText, AText : string) : Boolean
130: Function AnsiEndsText( const ASubText, AText : string) : Boolean
131: Function AnsiExtractQuotedStr( var Src : PChar; Quote : Char) : string
132: function AnsiExtractQuotedStr(var Src: PChar; Quote: Char): string)
133: Function AnsiIndexStr( const AText : string; const AValues : array of string) : Integer
134: Function AnsiIndexText( const AText : string; const AValues : array of string) : Integer
135: Function AnsiLastChar( S : string) : PChar
136: function AnsiLastChar(const S: string): PChar)
137: Function AnsiLeftStr( const AText : AnsiString; const ACount : Integer) : AnsiString
138: Function AnsiLowerCase( S : string) : string
139: Function AnsiLowercase(s : String) : String;
140: Function AnsiLowerCaseFileName( S : string) : string
141: Function AnsiMatchStr( const AText : string; const AValues : array of string) : Boolean
142: Function AnsiMatchText( const AText : string; const AValues : array of string) : Boolean
143: Function AnsiMidStr( const AText : AnsiString; const AStart, ACount : Integer) : AnsiString
144: Function AnsiPos( const src, sub : AnsiString) : Integer
145: Function AnsiPos( Substr, S : string) : Integer
146: function AnsiPos(const Substr: string; const S: string): Integer;)
147: Function AnsiQuotedStr( S : string; Quote : Char) : string
148: Function AnsiReplaceStr( const AText, AFromText, AToText : string) : string
149: Function AnsiReplaceText( const AText, AFromText, AToText : string) : string
150: Function AnsiResemblesText( const AText, AOther : string) : Boolean
151: Function AnsiReverseString( const AText : AnsiString) : AnsiString
152: Function AnsiRightStr( const AText : AnsiString; const ACount : Integer) : AnsiString
153: function AnsiSameCaption(const Text1: string; const Text2: string): Boolean)
154: Function AnsiSameStr( S1, S2 : string) : Boolean
155: function AnsiSameStr(const S1: string; const S2: string): Boolean)
156: Function AnsiSameText( const S1, S2 : string) : Boolean
157: Function AnsiSameText( S1, S2 : string) : Boolean
158: function AnsiSameText(const S1: string; const S2: string): Boolean)
159: Function AnsiStartsStr( const ASubText, AText : string) : Boolean
160: Function AnsiStartsText( const ASubText, AText : string) : Boolean
161: Function AnsiStrComp( S1, S2 : PChar) : Integer
162: function AnsiStrComp(S1: PChar; S2: PChar): Integer)
163: Function AnsiStrIComp( S1, S2 : PChar) : Integer
164: function AnsiStrIComp(S1: PChar; S2: PChar): Integer)
165: Function AnsiStrLastChar( P : PChar) : PChar
166: function AnsiStrLastChar(P: PChar): PChar)
167: Function AnsiStrLComp( S1, S2 : PChar; MaxLen : Cardinal) : Integer
168: Function AnsiStrLComp( S1, S2 : PChar; MaxLen : Cardinal) : Integer
169: Function AnsiStrLower( Str : PChar) : PChar
170: Function AnsiStrPos( Str, SubStr : PChar) : PChar
171: function AnsiStrPos(Str: PChar; SubStr: PChar): PChar)
172: Function AnsiStrScan(Str: PChar; Chr: Char): PChar)
173: Function AnsiStrUpper( Str : PChar) : PChar
174: Function AnsiToUtf8( const S : string) : UTF8String
175: Function AnsiToUtf8Ex( const S : string; const cp : integer) : UTF8String
176: Function AnsiUpperCase( S : string) : string
177: Function AnsiUppercase(s : String) : String;
178: Function AnsiUpperCaseFileName( S : string) : string

```

```

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

```

```

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

```



```

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

```

```

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

```

```

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

```

```

612: Function DelChar( const pStr : string; const pChar : Char) : string
613: Function DelEnvironmentVar( const Name : string) : Boolean
614: Function Delete( const MsgNum : Integer) : Boolean
615: Function DeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean) : Boolean
616: Function DeleteFile(const FileName: string): boolean
617: Function DeleteFileEx( FileName : string; Flags : FILEOP_FLAGS) : Boolean
618: Function DelimiterPosn( const sString : string; const sDelimiters: string): integer;
619: Function DelimiterPosn1(const sString:string;const sDelimiters:string;out cDelimiter: char) : integer;
620: Function DelSpace( const pStr : string) : string
621: Function DelString( const pStr, pDelStr : string) : string
622: Function DelTree( const Path : string) : Boolean
623: Function Depth : Integer
624: Function Description : string
625: Function DescriptionsAvailable : Boolean
626: Function DescriptionToConvFamily( const ADescription : string; out AFamily : TConvFamily) : Boolean
627: Function DescriptionToConvType( const ADescription : string; out AType : TConvType) : Boolean;
628: Function DescriptionToConvType1(const AFamil:TConvFamily;const ADescr:string;out AType:TConvType):Boolean;
629: Function DetectUTF8Encoding( const s : UTF8String) : TEncodeType
630: Function DialogsToPixelsX( const Dialogs : Word) : Word
631: Function DialogsToPixelsY( const Dialogs : Word) : Word
632: Function Digits( const X : Cardinal) : Integer
633: Function DirectoryExists( const Name : string) : Boolean
634: Function DirectoryExists( Directory : string) : Boolean
635: Function DiskFree( Drive : Byte) : Int64
636: function DiskFree(Drive: Byte): Int64
637: Function DiskInDrive( Drive : Char) : Boolean
638: Function DiskSize( Drive : Byte) : Int64
639: function DiskSize(Drive: Byte): Int64
640: Function DISPATCHCOMMAND( ACOMMAND : WORD) : BOOLEAN
641: Function DispatchEnabled : Boolean
642: Function DispatchMask : TMask
643: Function DispatchMethodType : TMethodType
644: Function DISPATCHPOPOP( 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 Executel(constCommText:WideString;const CType:TCommandType;const ExecuteOptions:TExecuteOptions):
_Recordset;
734: Function Executel( const Parameters : OleVariant ) : _Recordset;
735: Function Executel( ParentWnd : HWND ) : Boolean;
736: Function Execute2( var RecordsAffected : Integer; const Parameters : OleVariant ) : _Recordset;
737: Function ExecuteAction( Action : TBasicAction ) : Boolean
738: Function ExecuteDirect( const SQL : WideString ) : Integer
739: Function ExecuteFile(const FileName:string;const Params:string;const DefDir:string;ShowCmd:Int):THandle
740: Procedure ExecuteThread2(afunc:TThreadFunction2;thrOK:boolean);AddTypeS('TThreadFunction2','procedure
741: Function CreateThread2(ThreadFunc: TThreadFunction2) : THandle
742: function ExeFileIsRunning(ExeFile: string): boolean;
743: function ExePath: string;
744: function ExePathName: string;
745: Function Exists( AItem : Pointer ) : Boolean
746: Function ExitWindows( ExitCode : Cardinal ) : Boolean
747: function Exp(x: Extended): Extended;
748: Function ExpandEnvironmentVar( var Value : string ) : Boolean
749: Function ExpandFileName( FileName : string ) : string
750: function ExpandFileName(const FileName: string): string)
751: Function ExpandUNCFileName( FileName : string ) : string
752: function ExpandUNCFileName(const FileName: string): string)
753: Function ExpJ( const X : Float ) : Float;
754: Function Exsecans( X : Float ) : Float
755: Function Extract( const AByteCount : Integer ) : string
756: Function Extract( Item : TClass ) : TClass
757: Function Extract( Item : TComponent ) : TComponent
758: Function Extract( Item : TObject ) : TObject
759: Function ExtractFileDir( FileName : string ) : string
760: function ExtractFileDir(const FileName: string): string)
761: Function ExtractFileDrive( FileName : string ) : string
762: function ExtractFileDrive(const FileName: string): string)
763: Function ExtractFileExt( FileName : string ) : string
764: function ExtractFileExt(const FileName: string): string)
765: Function ExtractFileExtNoDot( const FileName : string ) : string
766: Function ExtractFileExtNoDotUpper( const FileName : string ) : string
767: Function ExtractFileName( FileName : string ) : string
768: function ExtractFileName(const filename: string):string;
769: Function ExtractFilePath( FileName : string ) : string
770: function ExtractFilePath(const filename: string):string;
771: Function ExtractRelativePath( BaseName, DestName : string ) : string
772: function ExtractRelativePath(const BaseName: string; const DestName: string): string)
773: Function ExtractShortPathName( FileName : string ) : string
774: function ExtractShortPathName(const FileName: string): string)
775: function ExtractStrings(Separators, WhiteSpace: TSysCharSet; Content: PChar;Strings: TStrings): Integer
776: function ExtractStrings(Separators:TSysCharSet;WhiteSpace:TSysCharSet;Content:PChar;Str:TStrings): Integer)
777: Function Fact(num: integer): Extended;
778: Function FactInt(num: integer): int64;
779: Function Factorial( const N : Integer ) : Extended
780: Function FahrenheitToCelsius( const AValue : Double ) : Double
781: function FalseBoolStrs: array of string
782: Function Fetch(var AInput:string;const ADelim:string;const ADelete:Bool;const ACaseSensitive:Bool):string
783: Function FetchCaseInsensitive(var AInput:string; const ADelim:string; const ADelete:Boolean): string
784: Function Fibo(num: integer): Extended;
785: Function FiboInt(num: integer): Int64;
786: Function Fibonacci( const N : Integer ) : Integer
787: Function FIELDBYNAME( const FIELDNAME : STRING ) : TFIELD

```

```

788: Function FIELDBYNAME( const FIELDNAME : WIDESTRING ) : TFIELD
789: Function FIELDBYNAME( const NAME : String ) : TFIELD
790: Function FIELDBYNAME( const NAME : String ) : TFIELDDEF
791: Function FIELDBYNUMBER( FIELDNO : INTEGER ) : TFIELD
792: Function FileAge( FileName : string ) : Integer
793: Function FileAge(const FileName: string): integer
794: Function FileCompareText( const A, B : String ) : Integer
795: Function FileContains(const FileName:string;Text:string;CaseSensitive:Bool;ExceptionOnError:Bool): Boolean
796: Function FileCreate( FileName : string ) : Integer;
797: Function FileCreate(const FileName: string): integer
798: Function FileCreateTemp( var Prefix : string ) : THandle
799: Function FileDateToDateTime( FileDate : Integer ) : TDateTime
800: function FileDateToDateTime(FileDate: Integer): TDateTime;
801: Function FileExists( const FileName : string ) : Boolean
802: Function FileExists( FileName : string ) : Boolean
803: function fileExists(const FileName: string): Boolean;
804: Function FileGetAttr( FileName : string ) : Integer
805: Function FileGetAttr(const FileName: string): integer
806: Function FileGetDate( Handle : Integer ) : Integer
807: Function FileGetDate(handle: integer): integer
808: Function FileGetDisplayName( const FileName : string ) : string
809: Function FileGetSize( const FileName : string ) : Integer
810: Function FileGetTempName( const Prefix : string ) : string
811: Function FileGetTypeNames( const FileName : string ) : string
812: Function FileIsReadOnly( FileName : string ) : Boolean
813: Function FileLoad( ResType : TResType; const Name : string; MaskColor : TColor ) : Boolean
814: Function FileOpen( FileName : string; Mode : LongWord ) : Integer
815: Function FileOpen(const FileName: string; mode:integer): integer
816: Function FileRead(handle: integer; Buffer: PChar; count: LongWord): integer
817: Function FileSearch( Name, DirList : string ) : string
818: Function FileSearch(const Name, dirList: string): string
819: Function FileSeek( Handle : Integer; Offset : Int64; Origin : Integer ) : Int64;
820: Function FileSeek( Handle, Offset, Origin : Integer ) : Integer;
821: Function FileSeek(handle, offset, origin: integer): integer
822: Function FileSetAttr( FileName : string; Attr : Integer ) : Integer
823: function FileSetAttr(const FileName: string; Attr: Integer): Integer;
824: Function FileSetDate(FileName : string; Age : Integer ) : Integer;
825: Function FileSetDate(handle: integer; age: integer): integer
826: Function FileSetDate2(FileHandle : Integer; Age : Integer ) : Integer;
827: Function FileSetDateH( Handle : Integer; Age : Integer ) : Integer;
828: Function FileSetReadOnly( FileName : string; ReadOnly : Boolean ) : Boolean
829: Function FileSize( const FileName : string ) : int64
830: Function FileSizeByName( const AFilename : string ) : Longint
831: function FileWrite(Handle: Integer; const Buffer: pChar; Count: LongWord): Integer;
832: Function FilterSpecArray : TComdlgFilterSpecArray
833: Function FIND( ACAPTION : String ) : TMENUITEM
834: Function Find( AItem : Pointer; out AData : Pointer ) : Boolean
835: Function FIND( const ANAME : String ) : TNAMEITEM
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 FindCmdLineSwitch( Switch : string ) : Boolean;
852: Function FindComponent(ANAME: String): TComponent;
853: function FindComponent(vlabel: string): TComponent;
854: function FindComponent2(vlabel: string): TComponent;
855: function FindControl(Handle: HWND): TWinControl;
856: Function FindData( StartIndex : Integer; Value : Pointer; Inclusive, Wrap : Boolean ) : TListItem
857: Function FindDatabase( const DatabaseName : string ) : TDatabase
858: function FindDragTarget(const Pos: TPoint; AllowDisabled: Boolean): TControl;
859: Function FINDFIELD( const FIELDNAME : STRING ) : TFIELD
860: Function FINDFIELD( const FIELDNAME : WideString ) : TFIELD
861: Function FindFirst2(const Path: string; Attr: Integer; var F: TSearchRec):Integer)
862: Function FindNext2(var F: TSearchRec): Integer)
863: procedure FindClose2(var F: TSearchRec)
864: Function FINDFIRST : BOOLEAN
865: TjvSpecialFolder = (sfRecycleBin, sfControlPanel, sfDesktop, sfDesktopDirectory,
866: sfMyComputer, sfFonts, sfNetHood, sfNetwork, sfPersonal, sfPrinters, sfPrograms,sfRecent,sfSendTo,
sfStartMenu, 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_SENTO, 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 -> hexabin! 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(aphat: 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 : TStringList; IncludeTrackType : Boolean; Drive : Char ) : string;
1017: Function GetCdInfo( InfoType : TJclCdMediaInfo; Drive : Char ) : string
1018: Function GetChangedText( const Text : string; SelStart, SelLength : Integer; Key : Char ) : string
1019: Function GetClientThread( ClientSocket : TServerClientWinSocket ) : TServerClientThread
1020: Function GetClockValue : Int64
1021: function getCmdLine: PChar;
1022: function getCmdShow: Integer;
1023: function GetCPUSpeed: Double;
1024: Function GetColField( DataCol : Integer ) : TField
1025: Function GetColorBlue( const Color : TColor ) : Byte
1026: Function GetColorFlag( const Color : TColor ) : Byte
1027: Function GetColorGreen( const Color : TColor ) : Byte
1028: Function GetColorRed( const Color : TColor ) : Byte
1029: Function GetComCtlVersion : Integer
1030: Function GetComPorts: TStringList;
1031: Function GetCommonAppdataFolder : string
1032: Function GetCommonDesktopdirectoryFolder : string
1033: Function GetCommonFavoritesFolder : string
1034: Function GetCommonFilesFolder : string
1035: Function GetCommonProgramsFolder : string
1036: Function GetCommonStartmenuFolder : string
1037: Function GetCommonStartupFolder : string
1038: Function GetComponent( Owner, Parent : TComponent ) : TComponent
1039: Function GetConnectionRegistryFile( DesignMode : Boolean ) : string
1040: Function GetCookiesFolder : string
1041: Function GetCPUSpeed( var CpuSpeed : TFreqInfo ) : Boolean
1042: Function GetCurrent : TFavoriteLinkItem
1043: Function GetCurrent : TListItem
1044: Function GetCurrent : TTaskDialogBaseButtonItem
1045: Function GetCurrent : TToolButton
1046: Function GetCurrent : TTreeNode
1047: Function GetCurrent : WideString
1048: Function GetCurrentDir : string

```



```

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

```

```

1138: Function GetIncome( const aNetto : Currency) : Currency
1139: Function GetIncome( const aNetto : Extended) : Extended
1140: Function GetIncome( const aNetto : Extended): Extended
1141: Function GetIncome(const aNetto : Extended) : Extended
1142: Function GetIncome(const aNetto: Currency): Currency
1143: Function GetIncome2( const aNetto : Currency) : Currency
1144: Function GetIncome2( const aNetto : Currency): Currency
1145: Function getIndex_Attrs( tag : string; var idx : Integer; var Attrs : string) : string
1146: Function GETINDEXFORFIELDS( const FIELDS : String; CASEINSENSITIVE : BOOLEAN) : TINDEXDEF
1147: Function GetIndexForOrderBy( const SQL : WideString; DataSet : TDataSet) : TIndexDef
1148: Function GetInstRes(Instance:THandle;ResType:TResType;const
    Name:string;Width:Integer;LoadFlags:TLoadResources;MaskColor:TColor):Boolean;
1149: Function
    GetInstRes1(Instance:THandle;ResType:TResType;ResID:DWORD;Width:Integer;LoadFlags:TLoadResources;MaskColor:
    TColor):Boolean;
1150: Function GetIntelCacheDescription( const D : Byte) : string
1151: Function GetInteractiveUserName : string
1152: Function GetInternetCacheFolder : string
1153: Function GetInternetFormattedFileTimeStamp( const AFilename : String) : String
1154: Function GetIPAddress( const HostName : string) : string
1155: Function GetIP( const HostName : string) : string
1156: Function GetIPHostByName(const AComputerName: String): String;
1157: Function GetIsAdmin: Boolean;
1158: Function GetItem( X, Y : Integer) : LongInt
1159: Function GetItemAt( X, Y : Integer) : TListItem
1160: Function GetItemHeight(Font: TFont): Integer;
1161: Function GetItemPath( Index : Integer) : string
1162: Function GetKeyFieldNames( List : 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: TStringList;
1241: Function GetProcessId : TIdPID
1242: Function GetProcessNameFromPid( PID : DWORD) : string
1243: Function GetProcessNameFromWnd( Wnd : HWND) : string
1244: Function GetProcessMemoryInfo(Process: THandle;ppsmemCounters: TProcessMemoryCounters;cb: DWORD):BOOL
1245: Function getProcessAllMemory(ProcessID : DWORD): TProcessMemoryCounters;
1246: Function getProcessMemoryInfo2(ProcessID : DWORD): TProcessMemoryCounters;
1247: Function GetProgramFilesFolder : string
1248: Function GetProgramsFolder : string
1249: Function GetProxy : string
1250: Function GetQuoteChar : WideString
1251: Function GetQrCode4(Width,Height:Word; Correct_Level:string;const
Data:string;aformat:string):TLinearBitmap;
1252: Function GetQrCodeToFile(Width,Height:Word;Correct_Level:string;const
Data:string;aformat:string):TLinearBitmap;
1253: Function GetRate : Double
1254: Function getPerfTime: string;
1255: Function getRuntime: string;
1256: Function GetRBitmap( Value : TBitmap) : TBitmap
1257: Function GetReadableName( const AName : string) : string
1258: Function GetRecentDocs : TStringList
1259: Function GetRecentFolder : string
1260: Function GetRecords( Count : Integer; out RecsOut : Integer; Options : Integer) : OleVariant;
1261: Function GetRecords1(Count:Integer; out RecsOut:Integer;Options:Integer;const CommandText:WideString;var
Params, OwnerData : OleVariant) : OleVariant;
1262: Function GetRecordset( const CommandText : WideString; ConnectionString : WideString) : _Recordset
1263: Function GetRegisteredCompany : string
1264: Function GetRegisteredOwner : string
1265: Function GetResource(ResType:TResType;const
Name:string;Width:Integer;LoadFlags:TLoadResources;MaskColor:TColor:Boolean
1266: Function GetResourceName( ObjStream : TStream; var AName : string) : Boolean
1267: Function GetResponse( const AAllowedResponses : array of SmallInt) : SmallInt;
1268: Function GetResponse1( const AAllowedResponse : SmallInt) : SmallInt;
1269: Function GetRValue( rgb : DWORD) : Byte
1270: Function GetGValue( rgb : DWORD) : Byte
1271: Function GetBValue( rgb : DWORD) : Byte
1272: Function GetCValue( cmyk : COLORREF) : Byte
1273: Function GetMValue( cmyk : COLORREF) : Byte
1274: Function GetYValue( cmyk : COLORREF) : Byte
1275: Function GetKValue( cmyk : COLORREF) : Byte
1276: Function CMYK( c, m, y, k : Byte) : COLORREF
1277: Function GetOSName: string;
1278: Function GetProcAddress( hModule : HMODULE; lpProcName : LPCSTR) : FARPROC
1279: Function GetProcAddress(Module: HMODULE; Proc: PChar): Dword
1280: Function GetSafeCallExceptionMsg : String
1281: Function GetSaturationBitmap( Value : TBitmap) : TBitmap
1282: Function GetSaveFilter : string
1283: Function GetSaver( Ext : string) : TBitmapLoader
1284: Function GetScrollLockKeyState : Boolean
1285: Function GetSearchString : string
1286: Function GetSelections( AList : TList) : TTreeNode
1287: Function GETSELTEXTBUF(BUFFER:PCHAR;BUFSIZE:INTEGER):INTEGER
1288: Function GetSendToFolder : string
1289: Function GetServer : IAppServer
1290: Function GetServerList : OleVariant
1291: Function GetShadowColor( const Color : TColor; Luminance : Integer) : TColor
1292: Function GetShellProcessHandle : THandle
1293: Function GetShellProcessName : string
1294: Function GetShellVersion : Cardinal
1295: Function getShortDayNames: string)
1296: Function GetShortHint(const hint: string): string
1297: Function getShortMonthNames: string)
1298: Function GetSizeOfFile( const FileName : string) : Int64;
1299: Function GetSizeOfFile1( Handle : THandle) : Int64;
1300: Function GetStdHandle(nStdHandle: Integer): Integer; stdcall;
1301: Function GetStartmenuFolder : string
1302: Function GetStartupFolder : string
1303: Function GetStringProperty( Instance : TPersistent; const PropName : string) : WideString
1304: Function GetSuccessor( cChar : char) : TniRegularExpressionState
1305: Function GetSwapFileSize : Integer
1306: Function GetSwapFileUsage : Integer
1307: Function GetSystemLocale : TIdCharSet
1308: Function GetSystemMetrics( nIndex : Integer) : Integer

```

```

1309: Function GetSystemPathSH(Folder: Integer): TFilename ;
1310: Function GetTableNameFromQuery( const SQL : WideString) : WideString
1311: Function GetTableNameFromSQL( const SQL : WideString) : WideString
1312: Function GetTableNameFromSQLEx( const SQL : WideString; IdOption : IDENTIFIEROption) : WideString
1313: Function GetTasksList( const List : TStringList) : Boolean
1314: Function getTeamViewerID: string;
1315: Function GetTemplatesFolder : string
1316: Function GetText : PwideChar
1317: function GetText:PChar
1318: Function GetTextBuf( Buffer : PChar; BufSize : Integer) : Integer
1319: function GETTEXTBUF(BUFFER:PCHAR;BUFSIZE:INTEGER):INTEGER
1320: Function GetTextItem( const Value : string) : Longint
1321: function GETTEXTLEN:INTEGER
1322: Function GetThreadLocale: Longint; stdcall
1323: Function GetCurrentThreadID: LongWord; stdcall;
1324: Function GetTickCount : Cardinal
1325: Function 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 HASCHILDEFS : BOOLEAN
1390: Function HasCurValues : Boolean
1391: Function HasExtendCharacter( const s : UTF8String) : Boolean
1392: Function HasFormat( Format : Word) : Boolean
1393: Function HashValue( AStream : TStream) : T5x4LongWordRecord;
1394: Function HashValue(AStream : TStream) : T4x4LongWordRecord
1395: Function HashValue(AStream: TStream): LongWord
1396: Function HashValue(AStream: TStream): Word
1397: Function HashValue1( AStream : TStream; const ABeginPos, AEndPos : Int64) : T5x4LongWordRecord;

```



```

1398: Function HashValue1(AStream : TStream): T4x4LongWordRecord
1399: Function HashValue128(const ASrc: string): T4x4LongWordRecord;
1400: Function HashValue128Stream(AStream: TStream): T4x4LongWordRecord;
1401: Function HashValue16(const ASrc: string): Word;
1402: Function HashValue16stream(AStream: TStream): Word;
1403: Function HashValue32(const ASrc: string): LongWord;
1404: Function HashValue32Stream(AStream: TStream): LongWord;
1405: Function HasMergeConflicts: Boolean
1406: Function hasMoreTokens: boolean
1407: Function HASPARENT: BOOLEAN
1408: Function HasParent: Boolean
1409: Function HasTransaction(Transaction: TDBXTransaction): Boolean
1410: Function HasUTF8BOM(S: TStream): boolean;
1411: Function HasUTF8BOM1(S: AnsiString): boolean;
1412: Function Haversine(X: Float): Float
1413: Function Head(s: string; const subs: string; var tail: string): string
1414: Function HELPCOMMAND(COMMAND: INTEGER; DATA: LONGINT): BOOLEAN
1415: Function HELPCONTEXT(CONTEXT: THELPCONTEXT): BOOLEAN
1416: Function HELPJUMP(JUMPID: STRING): BOOLEAN
1417: Function HeronianMean(const a, b: Float): Float
1418: Function HexStrToStr(Value: string): string;
1419: Function HexToBin(Text, Buffer: PChar; BufSize: Integer): Integer;
1420: Function HexToBin2(HexNum: string): string;
1421: Function HexToDouble(const Hex: string): Double
1422: Function HexToInt(hexnum: string): LongInt;
1423: Function HexToStr(Value: string): string;
1424: Function HexifyBlock(var Buffer, BufferSize: Integer): string
1425: Function Hi(vdat: word): byte;
1426: Function HiByte(W: Word): Byte)
1427: Function High: Int64;
1428: Function HighlightCell(DataCol, DataRow: Integer; const Value: string; AState: TGridDrawState): Boolean
1429: Function HINSTANCE: longword;
1430: Function HiWord(l: DWORD): Word)
1431: Function HMODULE: longword;
1432: Function HourOf(const AValue: TDateTime): Word
1433: Function HourOfDay(const AValue: TDateTime): Word
1434: Function HourOfMonth(const AValue: TDateTime): Word
1435: Function HourOfTheWeek(const AValue: TDateTime): Word
1436: Function HourOfTheYear(const AValue: TDateTime): Word
1437: Function HoursBetween(const ANow, ATen: TDateTime): Int64
1438: Function HourSpan(const ANow, ATen: TDateTime): Double
1439: Function HSLToRGB1(const H, S, L: Single): TColor32;
1440: Function HTMLDecode(const AStr: String): String
1441: Function HTMLEEncode(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 IBSMax(n1, n2: Integer): Integer
1448: Function IBSMin(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(aphash: string): string;
1469: Function IdHashSHA1(aphash: 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 : TJclFileMapView) : 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 IsDFASState : 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 : TStringList; ProcessID : DWORD; HandlesOnly : Boolean) : Boolean
1740: Function LoadFilesAsString( 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 MaskIntllLiteralToChar( 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 MilliSecondOfDay( 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 MinuteOfDay( 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 NEWBOTOMLINE : 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 NEWPOPMENU(OWNER:TCOMPONENT;const ANAME:String; ALIGNMENT:TPOUPALIGNMENT;AUTOPOPUP:BOOLEAN;
const ITEMS : array of TCMENUITEM) : TPOPMENU
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,syncoflag: 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 IsTCPPOpen(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, AHour, 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:TConvFam;const ADescr:string;const AFact:Double):TConvType
2194: Function RegistryRead(keyHandle: Longint; keyPath, myField: String) : string;
2195: Function ReleaseDC(hwnd: 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 : TStringList; 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_Rep1First( 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, ATen : TDateTime) : Int64
2355: Function SecondSpan( const ANow, ATen : 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 StringToArray( 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 StrParsel( 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, ATen : TDateTime) : Integer
2907: Function WeeksInAYear( const AYear : Word) : Word
2908: Function WeeksInYear( const AValue : TDateTime) : Word
2909: Function WeekSpan( const ANow, ATen : 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, ATen : TDateTime; const ADays : Integer) : Boolean
2946: Function WithinPastHours( const ANow, ATen : TDateTime; const AHours : Int64) : Boolean
2947: Function WithinPastMilliseconds( const ANow, ATen : TDateTime; const AMilliSeconds : Int64) : Boolean
2948: Function WithinPastMinutes( const ANow, ATen : TDateTime; const AMinutes : Int64) : Boolean
2949: Function WithinPastMonths( const ANow, ATen : TDateTime; const AMonths : Integer) : Boolean
2950: Function WithinPastSeconds( const ANow, ATen : TDateTime; const ASeconds : Int64) : Boolean
2951: Function WithinPastWeeks( const ANow, ATen : TDateTime; const AWeeks : Integer) : Boolean
2952: Function WithinPastYears( const ANow, ATen : 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 IntToWorldGridFormatIdent( 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:Integer;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,
  HighlightColor, ShadowColor: TColor; DrawHighlight: Boolean): TBitmap;
3015: function CreateDisabledBitmap(FOriginal: TBitmap; OutlineColor: TColor): TBitmap;
3016: function ChangeBitmapColor(Bitmap: TBitmap; Color, NewColor: TColor): TBitmap;
3017: procedure AssignBitmapCell(Source: TGraphic; Dest: TBitmap; Cols, Rows, Index: Integer);
3018: ($IFDEF WIN32)
3019: procedure ImageListDrawDisabled(Images: TImageList; Canvas: TCanvas;
  X, Y, Index: Integer; HighlightColor, GrayColor: TColor; DrawHighlight: Boolean);
3020: ($ENDIF)
3021: function MakeIcon(ResID: PChar): TIcon;
3022: function MakeIconID(ResID: Word): TIcon;
3023: function MakeModuleIcon(Module: THandle; ResID: PChar): TIcon;
3024: function CreateBitmapFromIcon(Icon: TIcon; BackColor: TColor): TBitmap;
3025: ($IFDEF WIN32)
3026: function CreateIconFromBitmap(Bitmap: TBitmap; TransparentColor: TColor): TIcon;
3027: ($ENDIF)
3028: { Service routines }
3029: procedure NotImplemented;
3030: procedure ResourceNotFound(ResID: PChar);
3031: function PointInRect(const P: TPoint; const R: TRect): Boolean;
3032: function PointInPolyRgn(const P: TPoint; const Points: array of TPoint): Boolean;
3033: function PaletteColor(Color: TColor): Longint;
3034: function WidthOf(R: TRect): Integer;
3035: function HeightOf(R: TRect): Integer;
3036: procedure PaintInverseRect(const RectOrg, RectEnd: TPoint);
3037: procedure DrawInvertFrame(ScreenRect: TRect; Width: Integer);
3038: procedure CopyParentImage(Control: TControl; Dest: TCanvas);
3039: procedure Delay(MSecs: Longint);
3040: procedure CenterControl(Control: TControl);
3041: Function PaletteEntries( Palette : HPALETTE) : Integer
3042: Function WindowClassName( Wnd : HWND) : string
3043: Function ScreenWorkArea : TRect
3044: Procedure MoveWindowOrg( DC : HDC; DX, DY : Integer)
3045: Procedure SwitchToWindow( Wnd : HWND; Restore : Boolean)
3046: Procedure ActivateWindow( Wnd : HWND)
3047: 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 MsgBox(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: {$ENDIF CBUILDER}
3071: function IsForegroundTask: Boolean;
3072: procedure MergeForm(AControl: TWinControl; AForm: TForm; Align: TAlign; Show: Boolean);
3073: function GetAveCharSize(Canvas: TCanvas): TPoint;
3074: function MinimizeText(const Text: string; Canvas: TCanvas; MaxWidth: Integer): string;
3075: procedure FreeUnusedOle;
3076: procedure Beep;
3077: function GetWindowsVersionJ: string;
3078: function LoadDLL(const LibName: string): THandle;
3079: function RegisterServer(const ModuleName: string): Boolean;
3080: {$IFDEF WIN32}
3081: function IsLibrary: Boolean;
3082: {$ENDIF}
3083: { Gradient filling routine }
3084: type TFillDirection = (fdTopToBottom, fdBottomToTop, fdLeftToRight, fdRightToLeft);
3085: procedure GradientFillRect(Canvas: TCanvas; ARect: TRect; StartColor, EndColor: TColor; Direction:
TFillDirection; Colors: Byte);
3086: { String routines }
3087: function GetEnvVar(const VarName: string): string;
3088: function AnsiUpperFirstChar(const S: string): string;
3089: function StringToPChar(var S: string): PChar;
3090: function StrPAlloc(const S: string): PChar;
3091: procedure SplitCommandLine(const CmdLine: string; var ExeName, Params: string);
3092: function DropT(const S: string): string;
3093: { Memory routines }
3094: function AllocMemo(Size: Longint): Pointer;
3095: function ReallocMemo(fpBlock: Pointer; Size: Longint): Pointer;
3096: procedure FreeMemo(var fpBlock: Pointer);
3097: function GetMemoSize(fpBlock: Pointer): Longint;
3098: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
3099: {$IFDEF COMPILER5_UP}
3100: procedure FreeAndNil(var Obj);
3101: {$ENDIF}
3102: // from PNGLoader
3103: Function OptimizeForPNG(Image:TLinearBitmap;QuantizationSteps:Integer;TransparentColor:TColor):Integer
3104: Procedure TransformRGB2LOCO( Image : TLinearBitmap)
3105: Procedure TransformLOCO2RGB( Image : TLinearBitmap)
3106: Procedure SortPalette( const Pal : TPalette; var ColorMap : TColorMap)
3107: Function DrawButtonFace( Canvas : TCanvas; const Client : TRect; BevelWidth : Integer; Style :
TButtonStyle; IsRounded, IsDown, IsFocused : Boolean) : TRect //TButtons
3108: AddDelphiFunction('Function IsAnAllResult( const AModalResult : TModalResult) : Boolean
3109: Function InitWndProc( HWindow : HWnd; Message : Longint; WParam : Longint; LParam : Longint) : Longint
3110: AddConstantN('CTL3D_ALL','LongWord').SetUInt($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', 'OnGetDefaultIniName'.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 : TObjet; const Section, Ident, Default : string) : string
3185: Procedure IniWriteString( IniFile : TObjet; const Section, Ident, Value : string)
3186: Function IniReadInteger( IniFile : TObjet; const Section, Ident : string; Default : Longint) : Longint
3187: Procedure IniWriteInteger( IniFile : TObjet; const Section, Ident : string; Value : Longint)
3188: Function IniReadBool( IniFile : TObjet; const Section, Ident : string; Default : Boolean) : Boolean
3189: Procedure IniWriteBool( IniFile : TObjet; const Section, Ident : string; Value : Boolean)
3190: Procedure IniReadSections( IniFile : TObjet; Strings : TStringList)
3191: Procedure IniEraseSection( IniFile : TObjet; const Section : string)
3192: Procedure IniDeleteKey( IniFile : TObjet; 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 : TObjet; const Section : string)
3197: Procedure InternalRestoreGridLayout( Grid : TCustomGrid; IniFile : TObjet; const Section : string)
3198: Procedure InternalSaveMDIChildren( MainForm : TForm; IniFile : TObjet)
3199: Procedure InternalRestoreMDIChildren( MainForm : TForm; IniFile : TObjet)
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 : TObjet; const Section : string)
3207: Procedure InternalRestoreFields(DataSet:TDataSet;IniFile:TObjet;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,erNullVariantErrorerInterfaceNotSupportedederError);
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 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 StrFillChar(const C: Char; Count: Integer): string'';
3451: Function IntFillChar(const I: Integer; Count: Integer): string'';
3452: Function ByteFillChar(const B: Byte; Count: Integer): string'';
3453: Function ArrFillChar(const AC: Char; Count: Integer): TCharArray;'';
3454: Function ArrByteFillChar(const AB: Char; Count: Integer): TByteArray;
3455: Function StrFind( const Substr, S : AnsiString; const Index : Integer ) : Integer
3456: ///Function StrHasPrefix( const S : AnsiString; const Prefixes : array of AnsiString ) : Boolean
3457: Function StrIndex( const S : AnsiString; const List : array of AnsiString ) : Integer
3458: Function StrILastPos( const SubStr, S : AnsiString ) : Integer
3459: Function StrIPos( const SubStr, S : AnsiString ) : Integer
3460: Function StrIsOneOf( const S : AnsiString; const List : array of AnsiString ) : Boolean
3461: Function StrLastPos( const SubStr, S : AnsiString ) : Integer
3462: Function StrMatch( const Substr, S : AnsiString; const Index : Integer ) : Integer
3463: Function StrMatches( const Substr, S : AnsiString; const Index : Integer ) : Boolean
3464: Function StrNIPos( const S, SubStr : AnsiString; N : Integer ) : Integer
3465: Function StrNPos( const S, SubStr : AnsiString; N : Integer ) : Integer
3466: Function StrPrefixIndex( const S : AnsiString; const Prefixes : array of AnsiString ) : Integer
3467: Function StrSearch( const Substr, S : AnsiString; const Index : Integer ) : Integer
3468: ///Function StrAfter( const SubStr, S : AnsiString ) : AnsiString
3469: ///Function StrBefore( const SubStr, S : AnsiString ) : AnsiString
3470: Function StrBetween( const S : AnsiString; const Start, Stop : AnsiChar ) : AnsiString
3471: Function StrChopRight( const S : AnsiString; N : Integer ) : AnsiString
3472: Function StrLeft( const S : AnsiString; Count : Integer ) : AnsiString
3473: Function StrMid( const S : AnsiString; Start, Count : Integer ) : AnsiString
3474: Function StrRestOf( const S : AnsiString; N : Integer ) : AnsiString
3475: Function StrRight( const S : AnsiString; Count : Integer ) : AnsiString
3476: Function CharEqualNoCase( const C1, C2 : AnsiChar ) : Boolean
3477: Function CharIsAlpha( const C : AnsiChar ) : Boolean
3478: Function CharIsAlphaNum( const C : AnsiChar ) : Boolean
3479: Function CharIsBlank( const C : AnsiChar ) : Boolean
3480: Function CharIsControl( const C : AnsiChar ) : Boolean
3481: Function CharIsDelete( const C : AnsiChar ) : Boolean
3482: Function CharIsDigit( const C : AnsiChar ) : Boolean
3483: Function CharIsLower( const C : AnsiChar ) : Boolean
3484: Function CharIsNumberChar( const C : AnsiChar ) : Boolean
3485: Function CharIsPrintable( const C : AnsiChar ) : Boolean
3486: Function CharIsPunctuation( const C : AnsiChar ) : Boolean
3487: Function CharIsReturn( const C : AnsiChar ) : Boolean
3488: Function CharIsSpace( const C : AnsiChar ) : Boolean
3489: Function CharIsUpper( const C : AnsiChar ) : Boolean

```



```

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

```

```

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

```

```

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

```

```

3749:   current Windows locale. The return value is the same as for CompareStr. }
3750: function AnsiStrLIComp(S1, S2: PChar; MaxLen: Cardinal): Integer;
3751: function AnsiStrIComp(S1, S2: PChar): Integer;
3752: ($ENDIF)
3753: function Var2Type(V: Variant; const VarType: Integer): Variant;
3754: function VarToInt(V: Variant): Integer;
3755: function VarToFloat(V: Variant): Double;
3756: { following functions are not documented because they are don't work properly , so don't use them }
3757: function ReplaceSokrl(S: string; const Word, Frase: string): string;
3758: { ReplaceSokrl is full equal to ReplaceString function - only for compatibility - don't use }
3759: { GetSubStr is full equal to SubStr function - only for compatibility - don't use }
3760: function GetSubStr(const S: string; const Index: Integer; const Separator: Char): string;
3761: function GetParameter: string;
3762: function GetLongFileName(FileName: string): string;
3763: (* from FileCtrl)
3764: function DirectoryExists(const Name: string): Boolean;
3765: procedure ForceDirectories(Dir: string);
3766: (# from FileCtrl)
3767: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
3768: function GetComputerID: string;
3769: function GetComputerName: string;
3770: {**** string routines }
3771: { ReplaceAllSokr searches for all substrings, Words,in a string, S, and replaces them with Frases with the
  same Index.Also see RAUtilsW.ReplaceSokrl function }
3772: function ReplaceAllSokr(S: string; Words, Frases: TStrings): string;
3773: { ReplaceSokr searches the Word in a string, S, on PosBeg position,
  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 }
3775: function ReplaceSokr(S:string;PosBeg,Len:Integer;Words,Frases:TStrings;var NewSelStart:Integer): string;
3776: { CountOfLines calculates the lines count in a string,each line separated from another with CrLf sequence }
3777: function CountOfLines(const S: string): Integer;
3778: { DeleteEmptyLines deletes all empty lines from strings, Ss. Lines contained only spaces also deletes. }
3779: procedure DeleteEmptyLines(Ss: TStrings);
3780: { SQLAddWhere adds or modifies existing where-statement, where, to the strings, SQL.
  Note: If strings SQL allready contains where-statement, it must be started on beginning of any line }
3782: procedure SQLAddWhere(SQL: TStrings; const Where: string);
3783: {**** files routines - }
3784: { ResSaveToFile save resource named as Name with Typ type into file FileName.
  Resource can be compressed using MS Compress program }
3786: function ResSaveToFile(const Typ,Name: string; const Compressed:Boolean; const FileName: string): Boolean;
3787: function ResSaveToFileEx(Inst:HINST;Typ,Name:PChar;const Compressed:Bool;const FileName:string): Bool
3788: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
3789: { Execute executes other program and waiting for it terminating, then return its Exit Code }
3790: function ExecuteJ(const CommandLine, WorkingDirectory: string): Integer;
3791: { IniReadSection read section, Section, from ini-file,
  IniFileName, into strings, Ss.This function reads ALL strings from specified section.
  Note: TIniFile.ReadSection function reads only strings with '=' symbol.}
3794: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
3795: { LoadTextFile load text file, FileName, into string }
3796: function LoadTextFile(const FileName: TFileName): string;
3797: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3798: { ReadFolder reads files list from disk folder, Folder, that are equal Mask, into strings, FileList}
3799: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
3800: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
3801: ($IFDEF COMPILER3_UP)
3802: { TargetFileName - if FileName is ShortCut returns filename ShortCut linked to }
3803: function TargetFileName(const FileName: TFileName): TFileName;
3804: { return filename ShortCut linked to }
3805: function ResolveLink(const hWnd: HWND; const LinkFile: TFileName; var FileName: TFileName): HRESULT;
3806: ($ENDIF COMPILER3_UP)
3807: {**** Graphic routines - }
3808: { LoadIcoToImage loads two icons from resource named NameRes,into two image lists ALarge and ASmall}
3809: procedure LoadIcoToImage(ALarge, ASmall: TImageList; const NameRes: string);
3810: { RAtextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
3811: procedure RAtextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
3812: { RAtextOutEx same with RAtextOut function, but can calculate needed height for correct output }
3813: function RAtextOutEx(Canvas:TCanvas; const R,RClip:TRect;const S: string;const CalcHeight:Boolean):Integer;
3814: { RAtextCalcHeight calculate needed height to correct output, using RAtextOut or RAtextOutEx functions }
3815: function RAtextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
3816: { Cinema draws some visual effect }
3817: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
3818: { Roughed fills rect with special 3D pattern }
3819: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
3820: { 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 }
3821: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
3822: { TextWidth calculate text with for writing using standard desktop font }
3823: function TextWidth(AStr: string): Integer;
3824: { DefineCursor load cursor from resource, and return available cursor number, assigned to it }
3825: function DefineCursor(Identifier: PChar): TCursor;
3826: {**** other routines - }
3827: { FindFormByClass returns first form specified class, FormClass,owned by Application global variable }
3828: function FindFormByClass(FormClass: TFormClass): TForm;
3829: function FindFormByClassName(FormClassName: string): TForm;
3830: { FindByTag returns the control with specified class, ComponentClass, from WinControl.Controls property,
  having Tag property value, equald to Tag parameter }
3832: function FindByTag(WinControl:TWinControl;ComponentClass:TComponentClass;const Tag:Integer):TComponent;
3833: { ControlAtPos2 equal to TWinControl.ControlAtPos function, but works better }
3834: function ControlAtPos2(Parent: TWinControl; X, Y: Integer): TControl;

```



```

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

```

```

3920: Procedure JvGetVarReference( S1, S2, SIdx : PAnsiChar)
3921: Procedure JvEatWhitespaceChars( S1 : PAnsiChar);
3922: Procedure JvEatWhitespaceChars1( S1 : PWideChar);
3923: Function GetTokenCount : Integer
3924: Procedure ResetTokenCount
3925: end;
3926:
3927: procedure SIRegister_JvDBQueryParamsForm(CL: TPSPascalCompiler);
3928: begin
3929:   SIRegister_TJvQueryParamsDialog(CL);
3930:   Function EditQueryParams( DataSet : TDataSet; List : TParams; AHelpContext : THelpContext ) : Boolean
3931: end;
3932:
3933: ***** JvStrUtil / JvStrUtils;*****
3934: function FindNotBlankCharPos(const S: string): Integer;
3935: function AnsiChangeCase(const S: string): string;
3936: function GetWordOnPos(const S: string; const P: Integer): string;
3937: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
3938: function Cmp(const S1, S2: string): Boolean;
3939: { Spaces returns string consists on N space chars }
3940: function Spaces(const N: Integer): string;
3941: { HasChar returns True, if char, Ch, contains in string, S }
3942: function HasChar(const Ch: Char; const S: string): Boolean;
3943: function HasAnyChar(const Chars: string; const S: string): Boolean;
3944: { SubStr returns substring from string, S, separated with Separator string}
3945: function SubStr(const S: string; const Index: Integer; const Separator: string): string;
3946: { SubStrEnd same to previous function but Index numerated from the end of string }
3947: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
3948: { ReplaceString searches for all substrings, OldPattern, in a string, S, replaces them with NewPattern }
3949: function ReplaceString(S: string; const OldPattern, NewPattern: string): string;
3950: function CharInSet(const Ch: Char; const SetOfChar: TSetOfChar): Boolean;
3951: { GetXYByPos is same to previous function, but returns X position in line too}
3952: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
3953: { AddSlash returns string with added slash char to Dir parameter, if needed }
3954: function AddSlash2(const Dir: TFileName): string;
3955: { AddPath returns FileName with Path, if FileName not contain any path }
3956: function AddPath(const FileName, Path: TFileName): TFileName;
3957: { ExePath returns ExtractFilePath(ParamStr(0)) }
3958: function ExePath: TFileName;
3959: function LoadTextFile(const FileName: TFileName): string;
3960: procedure SaveTextFile(const FileName: TFileName; const Source: string);
3961: { ConcatSep concatenate S and S2 strings with Separator. if S = '', separator don't included }
3962: function ConcatSep(const S, S2, Separator: string): string;
3963: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
3964: function FileEquMask(FileName, Mask: TFileName): Boolean;
3965: { FileEquMasks returns True if file, FileName, is compatible with given Masks.
3966:   Masks must be separated with comma (',';) }
3967: function FileEquMasks(FileName, Masks: TFileName): Boolean;
3968: function StringEndsWith(const Str, SubStr: string): Boolean;
3969: function ExtractFilePath2(const FileName: string): string;
3970: function StrToOem(const AnsiStr: string): string;
3971: { StrToOem translates a string from the Windows character set into the OEM character set. }
3972: function OemToAnsiStr(const OemStr: string): string;
3973: { OemToAnsiStr translates a string from the OEM character set into the Windows character set. }
3974: function IsEmptyStr(const S: string; const EmptyChars: TCharSet): Boolean;
3975: { EmptyStr returns true if the given string contains only character from the EmptyChars. }
3976: function ReplaceStr(const S, Srch, Replace: string): string;
3977: { Returns string with every occurrence of Srch string replaced with Replace string. }
3978: function DelSpace(const S: string): string;
3979: { DelSpace return a string with all white spaces removed. }
3980: function DelChars(const S: string; Chr: Char): string;
3981: { DelChars return a string with all Chr characters removed. }
3982: function DelBSpace(const S: string): string;
3983: { DelBSpace trims leading spaces from the given string. }
3984: function DelESpace(const S: string): string;
3985: { DelESpace trims trailing spaces from the given string. }
3986: function DelRSpace(const S: string): string;
3987: { DelRSpace trims leading and trailing spaces from the given string. }
3988: function DelSpace1(const S: string): string;
3989: { DelSpace1 return a string with all non-single white spaces removed. }
3990: function Tab2Space(const S: string; Numb: Byte): string;
3991: { Tab2Space converts any tabulation character in the given string to the Numb spaces characters. }
3992: function NPos(const C: string; S: string; N: Integer): Integer;
3993: { NPos searches for a N-th position of substring C in a given string. }
3994: function MakeStr(C: Char; N: Integer): string;
3995: function MS(C: Char; N: Integer): string;
3996: { MakeStr return a string of length N filled with character C. }
3997: function AddChar(C: Char; const S: string; N: Integer): string;
3998: { AddChar return a string left-padded to length N with characters C. }
3999: function AddCharR(C: Char; const S: string; N: Integer): string;
4000: { AddCharR return a string right-padded to length N with characters C. }
4001: function LeftStr(const S: string; N: Integer): string;
4002: { LeftStr return a string right-padded to length N with blanks. }
4003: function RightStr(const S: string; N: Integer): string;
4004: { RightStr return a string left-padded to length N with blanks. }
4005: function CenterStr(const S: string; Len: Integer): string;
4006: { CenterStr centers the characters in the string based upon the Len specified. }
4007: function CompStr(const S1, S2: string): Integer;
4008: {CompStr compares S1 to S2, case-sensitivity. return val is -1 if S1 < S2,0 if S1 = S2,or 1 if S1>S2. }

```

```

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

```

```

4097: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer;
4098:     {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4099:     {$IFDEF COMPILER4_UP}
4100: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4101:     {$ENDIF}
4102: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer;
4103:     {$IFDEF COMPILER4_UP} overload; {$ENDIF}
4104:     {$IFDEF COMPILER4_UP}
4105: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4106:     {$ENDIF}
4107: function GetTempDir: string;
4108: function GetWindowsDir: string;
4109: function GetSystemDir: string;
4110: function BrowseDirectory(var AFolderName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4111:     {$IFDEF WIN32}
4112: function BrowseComputer(var ComputerName:string; const DlgText:string; AHelpContext:THelpContext): Boolean;
4113: function ShortToLongFileName(const ShortName: string): string;
4114: function ShortToLongPath(const ShortName: string): string;
4115: function LongToShortFileName(const LongName: string): string;
4116: function LongToShortPath(const LongName: string): string;
4117: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4118: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4119:     {$ENDIF WIN32}
4120:     {$IFDEF COMPILER3_UP}
4121: function IsPathDelimiter(const S: string; Index: Integer): Boolean;
4122:     {$ENDIF}
4123: function CreateCalculatorForm( AOwner : TComponent; AHelpContext : THelpContext) : TJvCalculatorForm
4124: function CreatePopupCalculator( AOwner : TComponent; ABiDiMode : TBiDiMode) : TWinControl
4125: function CreatePopupCalculator( AOwner : TComponent) : TWinControl
4126: procedure SetupPopupCalculator( PopupCalc : TWinControl; APrecision : Byte; ABeepOnError : Boolean)
4127:
4128: //*****procedure SIRegister_VarHlpr(CL: TPSPascalCompiler);
4129: procedure VariantClear( var V : Variant)
4130: procedure VariantArrayRedim( var V : Variant; High : Integer)
4131: procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
4132: procedure VariantCpy( const src : Variant; var dst : Variant)
4133: procedure VariantAdd( const src : Variant; var dst : Variant)
4134: procedure VariantSub( const src : Variant; var dst : Variant)
4135: procedure VariantMul( const src : Variant; var dst : Variant)
4136: procedure VariantDiv( const src : Variant; var dst : Variant)
4137: procedure VariantMod( const src : Variant; var dst : Variant)
4138: procedure VariantAnd( const src : Variant; var dst : Variant)
4139: procedure VariantOr( const src : Variant; var dst : Variant)
4140: procedure VariantXor( const src : Variant; var dst : Variant)
4141: procedure VariantShl( const src : Variant; var dst : Variant)
4142: procedure VariantShr( const src : Variant; var dst : Variant)
4143: function VariantAdd2( const V1 : Variant; const V2 : Variant) : Variant
4144: function VariantSub2( const V1 : Variant; const V2 : Variant) : Variant
4145: function VariantMul2( const V1 : Variant; const V2 : Variant) : Variant
4146: function VariantDiv2( const V1 : Variant; const V2 : Variant) : Variant
4147: function VariantMod2( const V1 : Variant; const V2 : Variant) : Variant
4148: function VariantAnd2( const V1 : Variant; const V2 : Variant) : Variant
4149: function VariantOr2( const V1 : Variant; const V2 : Variant) : Variant
4150: function VariantXor2( const V1 : Variant; const V2 : Variant) : Variant
4151: function VariantShl2( const V1 : Variant; const V2 : Variant) : Variant
4152: function VariantShr2( const V1 : Variant; const V2 : Variant) : Variant
4153: function VariantNot( const V1 : Variant) : Variant
4154: function VariantNeg( const V1 : Variant) : Variant
4155: function VariantGetElement( const V : Variant; i1 : integer) : Variant;
4156: function VariantGetElement1( const V : Variant; i1, i2 : integer) : Variant;
4157: function VariantGetElement2( const V : Variant; i1, i2, i3 : integer) : Variant;
4158: function VariantGetElement3( const V : Variant; i1, i2, i3, i4 : integer) : Variant;
4159: function VariantGetElement4( const V : Variant; i1, i2, i3, i4, i5 : integer) : Variant;
4160: procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
4161: procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
4162: procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
4163: procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
4164: procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
4165: end;
4166:
4167: *****unit uPSI_JvgUtils;*****
4168: function IsEven(I: Integer): Boolean;
4169: function InchesToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4170: function CentimetersToPixels(DC: HDC; Value: Single; IsHorizontal: Boolean): Integer;
4171: procedure SwapInt(var I1, I2: Integer);
4172: function Spaces(Count: Integer): string;
4173: function DupStr(const Str: string; Count: Integer): string;
4174: function DupChar(C: Char; Count: Integer): string;
4175: procedure Msg(const AMsg: string);
4176: function RectW(R: TRect): Integer;
4177: function RectH(R: TRect): Integer;
4178: function IncColor(AColor: Longint; AOffset: Byte): Longint;
4179: function DecColor(AColor: Longint; AOffset: Byte): Longint;
4180: function IsItAFilledBitmap(Bmp: TBitmap): Boolean;
4181: procedure DrawTextInRectWithAlign(DC: HDC; R: TRect; const Text: string;
4182:     HAlign: TglHorAlign; VAlign: TglVertAlign; Style: TglTextStyle; Fnt: TFont; Flags: UINT);
4183: procedure DrawTextInRect(DC: HDC; R: TRect; const Text:string;Style:TglTextStyle;Fnt:TFont;Flags: UINT);
4184: procedure ExtTextOutExt(DC: HDC; X, Y: Integer; R: TRect; const Text: string;
4185:     Style: TglTextStyle; ADelineated, ASuppress3D: Boolean; FontColor, DelinColor, HighlightColor, ShadowColor:
TColor; Illumination: TJvgIllumination; Gradient: TJvgGradient; Font: TFont);

```



```

4186: procedure DrawBox(DC:HDC; var R:TRect; Style:TglBoxStyle;BackgrColor: Longint; ATransparent: Boolean);
4187: function DrawBoxEx(DC: HDC; ARect: TRect; Borders: TglSides;
4188:   BevelInner, BevelOuter: TPanelBevel; Bold:Boolean; BackgrColor: Longint;ATransparent: Boolean): TRect;
4189: procedure GradientBox(DC: HDC; R: TRect; Gradient: TJvgGradient;PenStyle, PenWidth: Integer);
4190: procedure ChangeBitmapColor(Bitmap: TBitmap; FromColor, ToColor: TColor);
4191: procedure DrawBitmapExt(DC: HDC; { DC - background & result}
4192:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; ///<...X,Y _in_ rect!
4193:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4194:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4195: procedure CreateBitmapExt(DC: HDC; { DC - background & result}
4196:   SourceBitmap: TBitmap; R: TRect; X, Y: Integer; ///<...X,Y _in_ rect!
4197:   BitmapOption: TglWallpaperOption; DrawState: TglDrawState;
4198:   ATransparent: Boolean; TransparentColor: TColor; DisabledMaskColor: TColor);
4199: procedure BringParentWindowToTop(Wnd: TWinControl);
4200: function GetParentForm(Control: TControl): TForm;
4201: procedure GetWindowImageFrom(Control:TWinControl;X,Y:Integer;ADrawSelf,ADrawChildWindows:Boolean;DC:HDC)
4202: procedure GetWindowImage(Control: TWinControl; ADrawSelf, ADrawChildWindows: Boolean; DC: HDC);
4203: procedure GetParentImageRect(Control: TControl; Rect: TRect; DC: HDC);
4204: function CreateRotatedFont(F: TFont; Escapement: Integer): HFONT;
4205: function FindMainWindow(const AWndClass, AWndTitle: string): THandle;
4206: procedure CalcShadowAndHighlightColors(BaseColor: TColor; Colors: TJvgLabelColors);
4207: function CalcMathString(AExpression: string): Single;
4208: function IIF(AExpression: Boolean; IfTrue, IfFalse: Variant): Variant; overload;
4209: function IIF(AExpression: Boolean; const IfTrue, IfFalse: string): string; overload;
4210: function GetTransparentColor(Bitmap: TBitmap; AutoTrColor: TglAutoTransparentColor): TColor;
4211: procedure TypeStringOnKeyboard(const S: string);
4212: function NextStringGridCell( Grid: TStringGrid ): Boolean;
4213: procedure DrawTextExtAligned(Canvas:TCanvas;const
  Text:string;R:TRect;Alignment:TglAlignment;WordWrap:Boolean);
4214: procedure LoadComponentFromTextFile(Component: TComponent; const FileName: string);
4215: procedure SaveComponentToTextFile(Component: TComponent; const FileName: string);
4216: function ComponentToString(Component: TComponent): string;
4217: procedure StringToComponent(Component: TComponent; const Value: string);
4218: function PlayWaveResource(const ResName: string): Boolean;
4219: function UserName: string;
4220: function ComputerName: string;
4221: function CreateIniFileName: string;
4222: function ExpandString(const Str: string; Len: Integer): string;
4223: function Transliterate(const Str: string; RusToLat: Boolean): string;
4224: function IsSmallFonts: Boolean;
4225: function SystemColorDepth: Integer;
4226: function GetFileTypeJ(const FileName: string): TglFileType;
4227: function GetFileType( hFile : THandle) : DWORD);
4228: function FindControlAtPt(Control: TWinControl; Pt: TPoint; MinClass: TClass): TControl;
4229: function StrPosExt(const Str1, Str2: PChar; Str2Len: DWORD): PChar;
4230:
4231: { *****Utility routines of unit classes}
4232: function LineStart(Buffer, BufPos: PChar): PChar
4233: function ExtractStrings(Separators, WhiteSpace: TSysCharSet; Content: PChar;'+
4234:   'Strings: TStrings): Integer
4235: function TestStreamFormat( Stream : TStream ) : TStreamOriginalFormat
4236: Procedure RegisterClass( AClass : TPersistentClass)
4237: Procedure RegisterClasses( AClasses : array of TPersistentClass)
4238: Procedure RegisterClassAlias( AClass : TPersistentClass; const Alias : string)
4239: Procedure UnRegisterClass( AClass : TPersistentClass)
4240: Procedure UnRegisterClasses( AClasses : array of TPersistentClass)
4241: Procedure UnRegisterModuleClasses( Module : HMODULE)
4242: Function FindGlobalComponent( const Name : string ) : TComponent
4243: Function IsUniqueGlobalComponentName( const Name : string ) : Boolean
4244: Function InitInheritedComponent( Instance : TComponent; RootAncestor : TClass ) : Boolean
4245: Function InitComponentRes( const ResName : string; Instance : TComponent ) : Boolean
4246: Function ReadComponentRes( const ResName : string; Instance : TComponent ) : TComponent
4247: Function ReadComponentResEx( HInstance : THandle; const ResName : string ) : TComponent
4248: Function ReadComponentResFile( const FileName : string; Instance : TComponent ) : TComponent
4249: Procedure WriteComponentResFile( const FileName : string; Instance : TComponent)
4250: Procedure GlobalFixupReferences
4251: Procedure GetFixupReferenceNames( Root : TComponent; Names : TStrings)
4252: Procedure GetFixupInstanceNames(Root: TComponent; const ReferenceRootName string; Names : TStrings)
4253: Procedure RedirectFixupReferences( Root : TComponent; const OldRootName, NewRootName : string)
4254: Procedure RemoveFixupReferences( Root : TComponent; const RootName : string)
4255: Procedure RemoveFixups( Instance : TPersistent)
4256: Function FindNestedComponent( Root : TComponent; const NamePath : string ) : TComponent
4257: Procedure BeginGlobalLoading
4258: Procedure NotifyGlobalLoading
4259: Procedure EndGlobalLoading
4260: Function GetUltimateOwner1( ACollection : TCollection ) : TPersistent;
4261: Function GetUltimateOwner( APersistent : TPersistent ) : TPersistent;
4262: // AddTypes('TWndMethod', 'Procedure (var Message : TMessage)
4263: //Function MakeObjectInstance( Method : TWndMethod ) : Pointer
4264: Procedure FreeObjectInstance( ObjectInstance : Pointer)
4265: // Function AllocateHWnd( Method : TWndMethod ) : Hwnd
4266: Procedure DeallocateHWnd( Wnd : Hwnd)
4267: Function AncestorIsValid( Ancestor : TPersistent; Root, RootAncestor : TComponent ) : Boolean
4268: *****unit uPSI_SqlTimSt and DB;*****
4269: Procedure VarSQLTimeStampCreate4( var aDest : Variant; const ASQLTimeStamp : TSQLTimeStamp);
4270: Function VarSQLTimeStampCreate3: Variant;
4271: Function VarSQLTimeStampCreate2( const AValue : string ) : Variant;
4272: Function VarSQLTimeStampCreatel( const AValue : TDateTime ) : Variant;
4273: Function VarSQLTimeStampCreate( const ASQLTimeStamp : TSQLTimeStamp ) : Variant;

```

```

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

```

```

4361: { $ENDIF MSWINDOWS }
4362: procedure DirFiles(const ADir, AMask: string; AFileList: TStringList);
4363: procedure RecurseDirFiles(const ADir: string; var AFileList: TStringList);
4364: procedure RecurseDirProgs(const ADir: string; var AFileList: TStringList);
4365: procedure SaveString(const AFile, AText: string);
4366: procedure SaveStringasFile(const AFile, AText: string);
4367: function LoadStringJ(const AFile: string): string;
4368: function LoadStringOfFile(const AFile: string): string;
4369: procedure SaveStringToFile(const AFile, AText: string);
4370: function LoadStringFromFile(const AFile: string): string;
4371: function HexToColor(const AText: string): TColor;
4372: function UppercaseHTMLTags(const AText: string): string;
4373: function LowercaseHTMLTags(const AText: string): string;
4374: procedure GetHTMLAnchors(const AFile: string; AList: TStringList);
4375: function RelativePath(const ASrc, ADst: string): string;
4376: function GetToken(var Start: Integer; const SourceText: string): string;
4377: function PosNonSpace(Start: Integer; const SourceText: string): Integer;
4378: function PosEscaped(Start: Integer; const SourceText, FindText: string; EscapeChar: Char): Integer;
4379: function DeleteEscaped(const SourceText: string; EscapeChar: Char): string;
4380: function BeginOfAttribute(Start: Integer; const SourceText: string): Integer;
4381: // parses the beginning of an attribute: space + alpha character
4382: function ParseAttribute(var Start: Integer; const SourceText: string; var AName, AValue: string): Boolean;
4383: // parses a name="value" attrib from Start; returns 0 when not found or else the position behind attribute
4384: procedure ParseAttributes(const SourceText: string; Attributes: TStringList);
4385: // parses all name=value attributes to the attributes TStringList
4386: function HasStrValue(const AText, AName: string; var AValue: string): Boolean;
4387: // checks if a name="value" pair exists and returns any value
4388: function GetStrValue(const AText, AName, ADefault: string): string;
4389: // retrieves string value from a line like:
4390: // name="jan verhoeven" email="janl dott verhoeven att wxs dott nl"
4391: // returns ADefault when not found
4392: function GetHTMLColorValue(const AText, AName: string; ADefault: TColor): TColor;
4393: // same for a color
4394: function GetIntValue(const AText, AName: string; ADefault: Integer): Integer;
4395: // same for an Integer
4396: function GetFloatValue(const AText, AName: string; ADefault: Extended): Extended;
4397: // same for a float
4398: function GetBoolValue(const AText, AName: string): Boolean;
4399: // same for Boolean but without default
4400: function GetValue(const AText, AName: string): string;
4401: // retrieves string value from a line like: name="jan verhoeven" email="janl verhoeven att wxs dott nl"
4402: procedure SetValue(var AText: string; const AName, AValue: string);
4403: // sets a string value in a line
4404: procedure DeleteValue(var AText: string; const AName: string);
4405: // deletes a AName="value" pair from AText
4406: procedure GetNames(AText: string; AList: TStringList);
4407: // get a list of names from a string with name="value" pairs
4408: function GetHTMLColor(AColor: TColor): string;
4409: // converts a color value to the HTML hex value
4410: function BackPosStr(Start: Integer; const FindString, SourceString: string): Integer;
4411: // finds a string backward case sensitive
4412: function BackPosText(Start: Integer; const FindString, SourceString: string): Integer;
4413: // finds a string backward case insensitive
4414: function PosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4415:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4416: // finds a text range, e.g. <TD>...</TD> case sensitive
4417: function PosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4418:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4419: // finds a text range, e.g. <TD>...</td> case insensitive
4420: function BackPosRangeStr(Start: Integer; const HeadString, TailString, SourceString: string;
4421:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4422: // finds a text range backward, e.g. <TD>...</TD> case sensitive
4423: function BackPosRangeText(Start: Integer; const HeadString, TailString, SourceString: string;
4424:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4425: // finds a text range backward, e.g. <TD>...</td> case insensitive
4426: function PosTag(Start: Integer; SourceString: string; var RangeBegin: Integer;
4427:   var RangeEnd: Integer): Boolean;
4428: // finds a HTML or XML tag: <....>
4429: function InnerTag(Start: Integer; const HeadString, TailString, SourceString: string;
4430:   var RangeBegin: Integer; var RangeEnd: Integer): Boolean;
4431: // finds the innertext between opening and closing tags
4432: function Easter(NYear: Integer): TDateTime;
4433: // returns the easter date of a year.
4434: function GetWeekNumber(Today: TDateTime): string;
4435: // gets a datecode. Returns year and weeknumber in format: YYWW
4436: function ParseNumber(const S: string): Integer;
4437: // parse number returns the last position, starting from 1
4438: function ParseDate(const S: string): Integer;
4439: // parse a SQL style data string from positions 1,
4440: // starts and ends with #
4441:
4442: *****unit JvJCLUtils;*****
4443:
4444: function VarIsInt(Value: Variant): Boolean;
4445: // VarIsInt returns VarIsOrdinal-[varBoolean]
4446: { PosIdx returns the index of the first appearance of SubStr in Str. The search starts at index "Index". }
4447: function PosIdx(const SubStr, S: string; Index: Integer = 0): Integer;
4448: function PosIdxW(const SubStr, S: WideString; Index: Integer = 0): Integer;
4449: function PosLastCharIdx(Ch: Char; const S: string; Index: Integer = 0): Integer;

```

```

4450: { GetWordOnPos returns Word from string, S, on the cursor position, P }
4451: function GetWordOnPos(const S: string; const P: Integer): string;
4452: function GetWordOnPosW(const S: WideString; const P: Integer): WideString;
4453: function GetWordOnPos2(const S: string; P: Integer; var iBeg, iEnd: Integer): string;
4454: function GetWordOnPos2W(const S: WideString; P: Integer; var iBeg, iEnd: Integer): WideString;
4455: { GetWordOnPosEx working like GetWordOnPos function, but
4456:   also returns Word position in iBeg, iEnd variables }
4457: function GetWordOnPosEx(const S: string; const P: Integer; var iBeg, iEnd: Integer): string;
4458: function GetWordOnPosExW(const S: WideString; const P: Integer; var iBeg, iEnd: Integer): WideString;
4459: function GetNextWordPosEx(const Text: string; StartIndex: Integer; var iBeg, iEnd: Integer): string;
4460: function GetNextWordPosExW(const Text: WideString; StartIndex: Integer; var iBeg, iEnd: Integer): WideString;
4461: procedure GetEndPosCaret(const Text: string; CaretX, CaretY: Integer; var X, Y: Integer);
4462: { GetEndPosCaret returns the caret position of the last char. For the position
4463:   after the last char of Text you must add 1 to the returned X value. }
4464: procedure GetEndPosCaretW(const Text: WideString; CaretX, CaretY: Integer; var X, Y: Integer);
4465: { GetEndPosCaret returns the caret position of the last char. For the position
4466:   after the last char of Text you must add 1 to the returned X value. }
4467: { SubStrBySeparator returns substring from string, S, separated with Separator string }
4468: function SubStrBySeparator(const S: string; const Index: Integer; const
  Separator: string; StartIndex: Integer = 1): string;
4469: function SubStrBySeparatorW(const S: WideString; const Index: Integer; const
  Separator: WideString; StartIndex: Integer = 1): WideString;
4470: { SubStrEnd same to previous function but Index numerated from the end of string }
4471: function SubStrEnd(const S: string; const Index: Integer; const Separator: string): string;
4472: { SubWord returns next Word from string, P, and offsets Pointer to the end of Word, P2 }
4473: function SubWord(P: PChar; var P2: PChar): string;
4474: function CurrencyByWord(Value: Currency): string;
4475: { GetLineByPos returns the Line number, there the symbol Pos is pointed. Lines separated with #13 symbol }
4476: function GetLineByPos(const S: string; const Pos: Integer): Integer;
4477: { GetXYByPos is same as GetLineByPos, but returns X position in line as well }
4478: procedure GetXYByPos(const S: string; const Pos: Integer; var X, Y: Integer);
4479: procedure GetXYByPosW(const S: WideString; const Pos: Integer; var X, Y: Integer);
4480: { ReplaceString searches for all substrings, OldPattern,
4481:   in a string, S, and replaces them with NewPattern }
4482: function ReplaceString(S: string; const OldPattern, NewPattern: string; StartIndex: Integer = 1): string;
4483: function ReplaceStringW(S: WideString; const OldPattern, NewPattern:
  WideString; StartIndex: Integer = 1): WideString;
4484: { ConcatSep concatenate S1 and S2 strings with Separator. if S = '' then separator not included }
4485: function ConcatSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
  SUPPORTS_INLINE}
4486: { ConcatLeftSep is same to previous function, but strings concatenate right to left }
4487: function ConcatLeftSep(const S1, S2, Separator: string): string; {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
  SUPPORTS_INLINE}
4488:
4489: { Next 4 function for russian chars transliterating.
4490:   This functions are needed because Oem2Ansi and Ansi2Oem functions sometimes suck }
4491: procedure Dos2Win(var S: AnsiString);
4492: procedure Win2Dos(var S: AnsiString);
4493: function Dos2WinRes(const S: AnsiString): AnsiString; inline; {$IFDEF SUPPORTS_INLINE}
4494: function Win2DosRes(const S: AnsiString): AnsiString; inline; {$IFDEF SUPPORTS_INLINE}
4495: function Win2Koi(const S: AnsiString): AnsiString;
4496: { FillString fills the string Buffer with Count Chars }
4497: procedure FillString(var Buffer: string; Count: Integer; const Value: Char); overload;
4498: procedure FillString(var Buffer: string; StartIndex, Count: Integer; const Value: Char); overload;
4499: { MoveString copies Count Chars from Source to Dest }
4500: procedure MoveString(const Source: string; var Dest: string; Count: Integer); {$IFDEF SUPPORTS_INLINE}
  inline; {$ENDIF SUPPORTS_INLINE} overload;
4501: procedure MoveString(const Source: string; SrcStartIdx: Integer; var Dest: string;
  DstStartIdx: Integer; Count: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE} overload;
4502: { FillWideChar fills Buffer with Count WideChars (2 Bytes) }
4503: procedure FillWideChar(var Buffer: string; Count: Integer; const Value: WideChar);
4504: { MoveWideChar copies Count WideChars from Source to Dest }
4505: procedure MoveWideChar(const Source: string; var Dest: string; Count: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF
  SUPPORTS_INLINE}
4506: { FillNativeChar fills Buffer with Count NativeChars }
4507: procedure FillNativeChar(var Buffer: string; Count: Integer; const Value: Char); // D2009 internal error {$IFDEF
  SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4508: { MoveNativeChar copies Count WideChars from Source to Dest }
4509: procedure MoveNativeChar(const Source: string; var Dest: string; Count: Integer); // D2009 internal error {$IFDEF
  SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4510: { IsSubString() compares the sub string to the string. Indices are 1th based. }
4511: function IsSubString(const S: string; StartIndex: Integer; const SubStr: string): Boolean;
4512: { Spaces returns string consists on N space chars }
4513: function Spaces(const N: Integer): string;
4514: { AddSpaces adds spaces to string S, if its Length is smaller than N }
4515: function AddSpaces(const S: string; const N: Integer): string;
4516: function SpacesW(const N: Integer): WideString;
4517: function AddSpacesW(const S: WideString; const N: Integer): WideString;
4518: { function LastDateRUS for russian users only }
4519: { returns date relative to current date: 'âââ âîý îâçââ' }
4520: function LastDateRUS(const Dat: TDateTime): string;
4521: { CurrencyToStr format Currency, Cur, using ffCurrency float format }
4522: function CurrencyToStr(const Cur: Currency): string;
4523: { HasChar returns True, if Char, Ch, contains in string, S }
4524: function HasChar(const Ch: Char; const S: string): Boolean;
4525: function HasCharW(const Ch: WideChar; const S: WideString): Boolean; inline; {$IFDEF SUPPORTS_INLINE}
4526: function HasAnyChar(const Chars: string; const S: string): Boolean;
4527: {$IFDEF COMPILER12_UP}
4528: function CharInSet(const Ch: AnsiChar; const SetOfChar: TSysCharSet): Boolean; inline; {$ENDIF SUPPORTS_INLINE}
4529:

```



```

4530: { $ENDIF ~COMPILER12_UP }
4531: function CharInSetW(const Ch: WideChar; const SetOfChar: TSysCharSet): Boolean; inline; { $ENDIF
SUPPORTS_INLINE }
4532: function CountOfChar(const Ch: Char; const S: string): Integer;
4533: function DefStr(const S: string; Default: string): string; { $IFDEF SUPPORTS_INLINE } inline; { $ENDIF
SUPPORTS_INLINE }
4534: { StrLICompW2 is a faster replacement for JclUnicode.StrLICompW }
4535: function StrLICompW2(S1, S2: PWideChar; MaxLen: Integer): Integer;
4536: function StrPosW(S, SubStr: PWideChar): PWideChar;
4537: function StrLenW(S: PWideChar): Integer;
4538: function TrimW(const S: WideString): WideString; { $IFDEF SUPPORTS_INLINE } inline; { $ENDIF SUPPORTS_INLINE }
4539: function TrimLeftW(const S: WideString): WideString; { $IFDEF SUPPORTS_INLINE } inline; { $ENDIF
SUPPORTS_INLINE }
4540: function TrimRightW(const S: WideString): WideString; inline; { $ENDIF SUPPORTS_INLINE }
4541: TPixelFormat = ( pfDevice, pf1bit, pf4bit, pf8bit, pf24bit )
4542: TMappingMethod = ( mmHistogram, mmQuantize, mmTrunc784, mmTrunc666, mmTripel, mmGrayscale )
4543: function GetBitmapPixelFormat( Bitmap : TBitmap ) : TPixelFormat
4544: function GetPaletteBitmapFormat( Bitmap : TBitmap ) : TPixelFormat
4545: procedure SetBitmapPixelFormat( Bitmap: TBitmap; PixelFormat: TPixelFormat; Method: TMappingMethod)
4546: function BitmapToMemoryStream(Bitmap: TBitmap; PixelFormat: TPixelFormat; Method: TMappingMethod): TMemoryStream;
4547: procedure GrayscaleBitmap( Bitmap : TBitmap)
4548: function BitmapToMemory( Bitmap : TBitmap; Colors : Integer ) : TStream
4549: procedure SaveBitmapToFile( const Filename : string; Bitmap : TBitmap; Colors : Integer)
4550: function ScreenPixelFormat : TPixelFormat
4551: function ScreenColorCount : Integer
4552: procedure TileImage( Canvas : TCanvas; Rect : TRect; Image : TGraphic)
4553: function ZoomImage( ImageW, ImageH, MaxW, MaxH : Integer; Stretch : Boolean ) : TPoint
4554: // SRegister_TJvGradient(CL);
4555:
4556: {***** files routines}
4557: procedure SetDelimitedText(List: TStrings; const Text: string; Delimiter: Char);
4558: const
4559: { $IFDEF MSWINDOWS }
4560: DefaultCaseSensitivity = False;
4561: { $ENDIF MSWINDOWS }
4562: { $IFDEF UNIX }
4563: DefaultCaseSensitivity = True;
4564: { $ENDIF UNIX }
4565: { GenTempFileName returns temporary file name on
drive, there FileName is placed }
4566: function GenTempFileName(FileName: string): string;
4567: { GenTempFileNameExt same to previous function, but
returning filename has given extension, FileExt }
4568: function GenTempFileNameExt(FileName: string; const FileExt: string): string;
4569: { ClearDir clears folder Dir }
4570: function ClearDir(const Dir: string): Boolean;
4571: { DeleteDir clears and than delete folder Dir }
4572: function DeleteDir(const Dir: string): Boolean;
4573: { FileEquMask returns True if file, FileName, is compatible with given dos file mask, Mask }
4574: function FileEquMask(FileMask: string; Mask: string; CaseSensitive: Boolean = DefaultCaseSensitivity): Boolean;
4575: { FileEqMasks returns True if file, FileName, is compatible with given Masks.
Masks must be separated with SepPath (MSW: ';' / UNIX: ':') }
4576: function FileEqMasks(FileMask: string; Masks: string): Boolean;
4577: CaseSensitive: Boolean = DefaultCaseSensitivity); Boolean;
4578: function DeleteFiles(const Folder: TFileName; const Masks: string): Boolean;
4579: { $IFDEF MSWINDOWS }
4580: { LZFileExpand expand file, FileSource,
into FileDest. Given file must be compressed, using MS Compress program }
4581: function LZFileExpand(const FileSource, FileDest: string): Boolean;
4582: { $ENDIF MSWINDOWS }
4583: { FileGetInfo fills SearchRec record for specified file attributes }
4584: function FileGetInfo(FileName: TFileName; var SearchRec: TSearchRec): Boolean;
4585: { HasSubFolder returns True, if folder APath contains other folders }
4586: function HasSubFolder(APath: TFileName): Boolean;
4587: { IsEmptyFolder returns True, if there are no files or
folders in given folder, APath }
4588: function IsEmptyFolder(APath: TFileName): Boolean;
4589: { AddSlash returns string with added slash Char to Dir parameter, if needed }
4590: function AddSlash(const Dir: TFileName): string; { $IFDEF SUPPORTS_INLINE } inline; { $ENDIF SUPPORTS_INLINE }
4591: { AddPath returns FileName with Path, if FileName not contain any path }
4592: function AddPath(const FileName, Path: TFileName): TFileName;
4593: function AddPaths(const PathList, Path: string): string;
4594: function ParentPath(const Path: TFileName): TFileName;
4595: function FindInPath(const FileName, PathList: string): TFileName;
4596: { DeleteReadOnlyFile clears R/O file attribute and delete file }
4597: function DeleteReadOnlyFile(const FileName: TFileName): Boolean;
4598: { HasParam returns True, if program running with specified parameter, Param }
4599: function HasParam(const Param: string): Boolean;
4600: function HasSwitch(const Param: string): Boolean;
4601: function Switch(const Param: string): string;
4602: { ExePath returns ExtractFilePath(ParamStr(0)) }
4603: function ExePath: TFileName; { $IFDEF SUPPORTS_INLINE } inline; { $ENDIF SUPPORTS_INLINE }
4604: function CopyDir(const SourceDir, DestDir: TFileName): Boolean;
4605: //function FileTimeToDateTime(const FT: TFileTime): TDateTime;
4606: procedure FileTimeToDosDateTimeWord(const FT: TFileTime; out Dft: DWORD);
4607: function MakeValidFileName(const FileName: TFileName; ReplaceBadChar: Char): TFileName;
4608: {*** Graphic routines }
4609: { IsTTFontSelected returns True, if True Type font is selected in specified device context }
4610: function IsTTFontSelected(const DC: HDC): Boolean;

```

```

4616: function KeyPressed(VK: Integer): Boolean;
4617: Function isKeyPressed: boolean; //true if key on memo2 (shell output) is pressed
4618: { TrueInflateRect inflates rect in other method, than InflateRect API function }
4619: function TrueInflateRect(const R: TRect; const I: Integer): TRect;
4620: {**** Color routines }
4621: procedure RGBToHSV(R, G, B: Integer; var H, S, V: Integer);
4622: function RGBToBGR(Value: Cardinal): Cardinal;
4623: //function ColorToPrettyName(Value: TColor): string;
4624: //function PrettyNameToColor(const Value: string): TColor;
4625: {**** other routines }
4626: procedure SwapInt(var Int1, Int2: Integer); {$IFDEF SUPPORTS_INLINE} inline; {$ENDIF SUPPORTS_INLINE}
4627: function IntPower(Base, Exponent: Integer): Integer;
4628: function ChangeTopException(E: TObject): TObject; // Linux version writes error message to ErrOutput
4629: function StrToBool(const S: string): Boolean;
4630: function Var2Type(V: Variant; const DestVarType: Integer): Variant;
4631: function VarToInt(V: Variant): Integer;
4632: function VarToFloat(V: Variant): Double;
4633: { following functions are not documented because they not work properly sometimes, so do not use them }
4634: // (rom) ReplaceStrings1, GetSubStr removed
4635: function GetLongFileName(const FileName: string): string;
4636: function FileNewExt(const FileName, NewExt: TFileName): TFileName;
4637: function GetParameter: string;
4638: function GetComputerID: string;
4639: function GetComputerName: string;
4640: {**** string routines }
4641: { ReplaceAllStrings searches for all substrings, Words,
4642:   in a string, S, and replaces them with Frases with the same Index. }
4643: function ReplaceAllStrings(const S: string; Words, Frases: TStrings): string;
4644: { ReplaceStrings searches the Word in a string, S, on PosBeg position,
4645:   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 }
4646: function ReplaceStrings(const S: string; PosBeg, Len: Int; Words, Frases: TStrings; var NewSelStart: Int): string;
4647: { CountOfLines calculates the lines count in a string, S,
4648:   each line must be separated from another with CrLf sequence }
4649: function CountOfLines(const S: string): Integer;
4650: { DeleteLines deletes all lines from strings which in the words, words.
4651:   The word of will be deleted from strings. }
4652: procedure DeleteOfLines(Ss: TStrings; const Words: array of string);
4653: { DeleteEmptyLines deletes all empty lines from strings, Ss.
4654:   Lines contained only spaces also deletes. }
4655: procedure DeleteEmptyLines(Ss: TStrings);
4656: { SQLAddWhere addes or modifies existing where-statement, where,
4657:   to the strings, SQL. Note: If strings SQL already contains where-statement,
4658:   it must be started on the begining of any line }
4659: procedure SQLAddWhere(SQL: TStrings; const Where: string);
4660: {**** files routines - }
4661: {$IFDEF MSWINDOWS}
4662: { ResSaveToFile save resource named as Name with Typ type into file FileName.
4663:   Resource can be compressed using MS Compress program}
4664: function ResSaveToFile(const Typ, Name: string; const Compressed: Boolean; const FileName: string): Boolean;
4665: function ResSaveToFileEx(Instance: HINST; Typ, Name: PChar; const Compressed: Boolean; const FileName: string):
   Boolean;
4666: function ResSaveToString(Instance: HINST; const Typ, Name: string; var S: string): Boolean;
4667: {$ENDIF MSWINDOWS}
4668: { IniReadSection read section, Section, from ini-file,
4669:   IniFileName, into strings, Ss. This function reads ALL strings from specified section.
4670:   Note: TIniFile.ReadSection function reads only strings with '=' symbol.}
4671: function IniReadSection(const IniFileName: TFileName; const Section: string; Ss: TStrings): Boolean;
4672: { LoadTextFile load text file, FileName, into string }
4673: function LoadTextFile(const FileName: TFileName): string;
4674: procedure SaveTextFile(const FileName: TFileName; const Source: string);
4675: { ReadFolder reads files list from disk folder, Folder, that are equal to mask, Mask, into strings, FileList}
4676: function ReadFolder(const Folder, Mask: TFileName; FileList: TStrings): Integer;
4677: function ReadFolders(const Folder: TFileName; FolderList: TStrings): Integer;
4678: { RTextOut same with TCanvas.TextOut procedure, but can clipping drawing with rectangle, RClip. }
4679: procedure RTextOut(Canvas: TCanvas; const R, RClip: TRect; const S: string);
4680: { RTextOutEx same with RTextOut function, but can calculate needed height for correct output }
4681: function RTextOutEx(Canvas: TCanvas; const R, RClip: TRect; const S: string; const CalcHeight: Boolean): Integer;
4682: { RTextCalcHeight calculate needed height for
4683:   correct output, using RTextOut or RTextOutEx functions }
4684: function RTextCalcHeight(Canvas: TCanvas; const R: TRect; const S: string): Integer;
4685: { Cinema draws some visual effect }
4686: procedure Cinema(Canvas: TCanvas; rS {Source}, rD {Dest}: TRect);
4687: { Roughed fills rect with special 3D pattern }
4688: procedure Roughed(ACanvas: TCanvas; const ARect: TRect; const AVert: Boolean);
4689: { 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 }
4690: function BitmapFromBitmap(SrcBitmap: TBitmap; const AWidth, AHeight, Index: Integer): TBitmap;
4691: { TextWidth calculate text width for writing using standard desktop font }
4692: function TextWidth(const AStr: string): Integer;
4693: { TextHeight calculate text height for writing using standard desktop font }
4694: function TextHeight(const AStr: string): Integer;
4695: procedure SetChildPropOrd(Owner: TComponent; const PropName: string; Value: Longint);
4696: procedure Error(const Msg: string);
4697: procedure ItemHtDrawEx(Canvas: TCanvas; Rect: TRect; const State: TOwnerDrawState; const Text: string;
   const HideSelColor: Boolean; var PlainItem: string; var Width: Integer; CalcWidth: Boolean);
4698: {example Text parameter: 'Item 1<b>bold</b><i>italic ITALIC <c:Red>red<c:Green>green<c:blue>blue</i>' }
4700: function ItemHtDraw(Canvas: TCanvas; Rect: TRect;
4701:   const State: TOwnerDrawState; const Text: string; const HideSelColor: Boolean): string;

```

```

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

```

```

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

```



```

4879: { Dec2Hex converts the given value to a hexadecimal string representation
4880:   with the minimum number of digits (A) specified. }
4881: function Hex2Dec(const S: string): Longint;
4882: { Hex2Dec converts the given hexadecimal string to the corresponding integer value. }
4883: function Dec2Numb(N: Int64; A, B: Byte): string;
4884: { Dec2Numb converts the given value to a string representation with the
4885:   base equal to B and with the minimum number of digits (A) specified. }
4886: function Numb2Dec(S: string; B: Byte): Int64;
4887: { Numb2Dec converts the given B-based numeric string to the corresponding
4888:   integer value. }
4889: function IntToBin(Value: Longint; Digits, Spaces: Integer): string;
4890: { IntToBin converts the given value to a binary string representation
4891:   with the minimum number of digits specified. }
4892: function IntToRoman(Value: Longint): string;
4893: { IntToRoman converts the given value to a roman numeric string representation. }
4894: function RomanToInt(const S: string): Longint;
4895: { RomanToInt converts the given string to an integer value. If the string
4896:   doesn't contain a valid roman numeric value, the 0 value is returned. }
4897: function FindNotBlankCharPos(const S: string): Integer;
4898: function FindNotBlankCharPosW(const S: WideString): Integer;
4899: function AnsiChangeCase(const S: string): string;
4900: function WideChangeCase(const S: string): string;
4901: function StartsText(const SubStr, S: string): Boolean;
4902: function EndsText(const SubStr, S: string): Boolean;
4903: function DequotedStr(const S: string; QuoteChar: Char = ''' ): string;
4904: function AnsiDequotedStr(const S: string; AQuote: Char): string; //follow Delphi 2009's "Ansi" prefix
4905: {end JvStrUtils}
4906: {$IFDEF UNIX}
4907: function GetTempFileName(const Prefix: AnsiString): AnsiString;
4908: {$ENDIF UNIX}
4909: { begin JvFileUtil }
4910: function FileDateTime(const FileName: string): TDateTime;
4911: function HasAttr(const FileName: string; Attr: Integer): Boolean;
4912: function DeleteFilesEx(const FileMasks: array of string): Boolean;
4913: function NormalDir(const DirName: string): string;
4914: function RemoveBackSlash(const DirName: string): string; // only for Windows/DOS Paths
4915: function ValidFileName(const FileName: string): Boolean;
4916: {$IFDEF MSWINDOWS}
4917: function FileLock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4918: function FileLock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4919: function FileUnlock(Handle: Integer; Offset, LockSize: Longint): Integer; overload;
4920: function FileUnlock(Handle: Integer; Offset, LockSize: Int64): Integer; overload;
4921: {$ENDIF MSWINDOWS}
4922: function GetWindowsDir: string;
4923: function GetSystemDir: string;
4924: function ShortToLongFileName(const ShortName: string): string;
4925: function LongToShortFileName(const LongName: string): string;
4926: function ShortToLongPath(const ShortName: string): string;
4927: function LongToShortPath(const LongName: string): string;
4928: {$IFDEF MSWINDOWS}
4929: procedure CreateFileLink(const FileName, DisplayName: string; Folder: Integer);
4930: procedure DeleteFileLink(const DisplayName: string; Folder: Integer);
4931: {$ENDIF MSWINDOWS}
4932: { end JvFileUtil }
4933: // Works like PtInRect but includes all edges in comparision
4934: function PtInRectInclusive(R: TRect; Pt: TPoint): Boolean;
4935: // Works like PtInRect but excludes all edges from comparision
4936: function PtInRectExclusive(R: TRect; Pt: TPoint): Boolean;
4937: function FourDigitYear: Boolean; {$IFDEF SUPPORTS_DEPRECATED} deprecated; {$ENDIF}
4938: function IsFourDigitYear: Boolean;
4939: { moved from JvJCLUtils }
4940: //Open an object with the shell (url or something like that)
4941: function OpenObject(const Value: string): Boolean; overload;
4942: function OpenObject(Value: PChar): Boolean; overload;
4943: {$IFDEF MSWINDOWS}
4944: //Raise the last Exception
4945: procedure RaiseLastWin32; overload;
4946: procedure RaiseLastWin32(const Text: string); overload;
4947: //Raise the last Exception with a small comment from your part { GetFileVersion returns the most
significant 32 bits of a file's binary version number. Typically, this includes the major and minor
version placed together in one 32-bit Integer. I
4948: function GetFileVersion(const AFileName: string): Cardinal;
4949: {$EXTERNALSYM GetFileVersion}
4950: //Get version of Shell.dll
4951: function GetShellVersion: Cardinal;
4952: {$EXTERNALSYM GetShellVersion}
4953: // CD functions on HW
4954: procedure OpenCdDrive;
4955: procedure CloseCdDrive;
4956: // returns True if Drive is accessible
4957: function DiskInDrive(Drive: Char): Boolean;
4958: {$ENDIF MSWINDOWS}
4959: //Same as linux function ;)
4960: procedure PError(const Text: string);
4961: // execute a program without waiting
4962: procedure Exec(const FileName, Parameters, Directory: string);
4963: // execute a program and wait for it to finish
4964: function ExecuteAndWait(CmdLine: string; const WorkingDirectory: string; Visibility: Integer = SW_SHOW): Int;
4965: // returns True if this is the first instance of the program that is running

```

```

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

```

```

5052: procedure FreeMemo(var fpBlock: Pointer);
5053: function GetMemoSize(fpBlock: Pointer): Longint;
5054: function CompareMem(fpBlock1, fpBlock2: Pointer; Size: Cardinal): Boolean;
5055: { Manipulate huge pointers routines }
5056: procedure HugeInc(var HugePtr: Pointer; Amount: Longint);
5057: procedure HugeDec(var HugePtr: Pointer; Amount: Longint);
5058: function HugeOffset(HugePtr: Pointer; Amount: Longint): Pointer;
5059: procedure HugeMove(Base: Pointer; Dst, Src, Size: Longint);
5060: procedure HMemCpy(DstPtr, SrcPtr: Pointer; Amount: Longint);
5061: function WindowClassName(Wnd: THandle): string;
5062: procedure SwitchToWindow(Wnd: THandle; Restore: Boolean);
5063: procedure ActivateWindow(Wnd: THandle);
5064: procedure ShowWinNoAnimate(Handle: THandle; CmdShow: Integer);
5065: procedure KillMessage(Wnd: THandle; Msg: Cardinal);
5066: { SetWindowTop put window to top without recreating window }
5067: procedure SetWindowTop(const Handle: THandle; const Top: Boolean);
5068: procedure CenterWindow(Wnd: THandle);
5069: function MakeVariant(const Values: array of Variant): Variant;
5070: { Convert dialog units to pixels and backwards }
5071: { $IFDEF MSWINDOWS }
5072: function DialogUnitsToPixelsX(DlgUnits: Word): Word;
5073: function DialogUnitsToPixelsY(DlgUnits: Word): Word;
5074: function PixelsToDialogUnitsX(PixUnits: Word): Word;
5075: function PixelsToDialogUnitsY(PixUnits: Word): Word;
5076: { $ENDIF MSWINDOWS }
5077: function GetUniqueFileNameInDir(const Path, FileNameMask: string): string;
5078: { $IFDEF BCB }
5079: function FindPrevInstance(const MainFormClass: ShortString; const ATitle: string): THandle;
5080: function ActivatePrevInstance(const MainFormClass: ShortString; const ATitle: string): Boolean;
5081: { $ELSE }
5082: function FindPrevInstance(const MainFormClass, ATitle: string): THandle;
5083: function ActivatePrevInstance(const MainFormClass, ATitle: string): Boolean;
5084: { $ENDIF BCB }
5085: { $IFDEF MSWINDOWS }
5086: { BrowseForFolderNative displays Browse For Folder dialog }
5087: function BrowseForFolderNative(const Handle: THandle; const Title: string; var Folder: string): Boolean;
5088: { $ENDIF MSWINDOWS }
5089: procedure AntiAlias(Clip: TBitmap);
5090: procedure AntiAliasRect(Clip: TBitmap; XOrigin, YOrigin, XFinal, YFinal: Integer);
5091: procedure CopyRectDIBits(ACanvas: TCanvas; const DestRect: TRect;
5092:   ABitmap: TBitmap; const SourceRect: TRect);
5093: function IsTrueType(const FontName: string): Boolean;
5094: // Removes all non-numeric characters from AValue and returns the resulting string
5095: function TextToValText(const AValue: string): string;
5096: function ExecRegExpr(const ARegExpr, AInputStr: RegExprString): boolean
5097: procedure SplitRegExpr(const ARegExpr, AInputStr: RegExprString; APieces: TStrings)
5098: function ReplaceRegExpr(const ARegExpr, AInputStr,
5099:   AReplaceStr: RegExprString; AUseSubstitution: bool): RegExprString;
5099: function QuoteRegExprMetaChars(const AStr: RegExprString): RegExprString
5100: function RegExprSubExpressions(const ARegExpr: string; ASubExprs: TStrings; AExtendedSyntax: boolean):
5101:
5102: *****unit uPSI_JvTFUtils;
5103: function JExtractYear( ADate: TDateTime): Word
5104: function JExtractMonth( ADate: TDateTime): Word
5105: function JExtractDay( ADate: TDateTime): Word
5106: function ExtractHours( ATime: TDateTime): Word
5107: function ExtractMins( ATime: TDateTime): Word
5108: function ExtractSecs( ATime: TDateTime): Word
5109: function ExtractMSecs( ATime: TDateTime): Word
5110: function FirstOfMonth( ADate: TDateTime): TDateTime
5111: function GetDayOfNthDOW( Year, Month, DOW, N: Word): Word
5112: function GetWeeksInMonth( Year, Month: Word; StartOfWeek: Integer): Word
5113: procedure IncBorlDOW( var BorlDOW: Integer; N: Integer)
5114: procedure IncDOW( var DOW: TTFDayOfWeek; N: Integer)
5115: procedure IncDays( var ADate: TDateTime; N: Integer)
5116: procedure IncWeeks( var ADate: TDateTime; N: Integer)
5117: procedure IncMonths( var ADate: TDateTime; N: Integer)
5118: procedure IncYears( var ADate: TDateTime; N: Integer)
5119: function EndOfMonth( ADate: TDateTime): TDateTime
5120: function IsFirstOfMonth( ADate: TDateTime): Boolean
5121: function IsEndOfMonth( ADate: TDateTime): Boolean
5122: procedure EnsureMonth( Month: Word)
5123: procedure EnsureDOW( DOW: Word)
5124: function EqualDates( D1, D2: TDateTime): Boolean
5125: function Lesser( N1, N2: Integer): Integer
5126: function Greater( N1, N2: Integer): Integer
5127: function GetDivLength( TotalLength, DivCount, DivNum: Integer): Integer
5128: function GetDivNum( TotalLength, DivCount, X: Integer): Integer
5129: function GetDivStart( TotalLength, DivCount, DivNum: Integer): Integer
5130: function DOWToBorl( ADOW: TTFDayOfWeek): Integer
5131: function BorlToDOW( BorlDOW: Integer): TTFDayOfWeek
5132: function DateToDOW( ADate: TDateTime): TTFDayOfWeek
5133: procedure CalcTextPos( HostRect: TRect; var TextLeft, TextTop: Integer; var TextBounds: TRect;
5134:   AFont: TFont; AAngle: Integer; HAlign: TAlignment; VAlign: TJvTFVAlignment; ATxt: string)
5134: procedure DrawAngleText( ACanvas: TCanvas; HostRect: TRect; var TextBounds: TRect; AAngle: Integer;
5135:   HAlign: TAlignment; VAlign: TJvTFVAlignment; ATxt: string)
5135: function JRectWidth( ARect: TRect): Integer
5136: function JRectHeight( ARect: TRect): Integer
5137: function JEmptyRect: TRect

```

```

5138: Function IsClassByName( Obj : TObject; ClassName : string) : Boolean
5139:
5140: procedure SIRegister_MSysUtils(CL: TPSPascalCompiler);
5141: begin
5142:   Procedure HideTaskBarButton( hWnd : HWND)
5143:   Function msLoadStr( ID : Integer) : String
5144:   Function msFormat( fmt : String; params : array of const) : String
5145:   Function msFileExists( const FileName : String) : Boolean
5146:   Function msIntToStr( Int : Int64) : String
5147:   Function msStrPas( const Str : PChar) : String
5148:   Function msRenameFile( const OldName, NewName : String) : Boolean
5149:   Function CutFileName( s : String) : String
5150:   Function GetVersionInfo( var VersionString : String) : DWORD
5151:   Function FormatTime( t : Cardinal) : String
5152:   Function msCreateDir( const Dir : string) : Boolean
5153:   Function SetAutoRun( NeedAutoRun : Boolean; AppName : String) : Boolean
5154:   Function SetTreeViewStyle( const hTV : HWND; dwNewStyle : dword) : DWORD
5155:   Function msStrLen( Str : PChar) : Integer
5156:   Function msDirectoryExists( const Directory : String) : Boolean
5157:   Function GetFolder( hWnd : hWnd; RootDir : Integer; Caption : String) : String
5158:   Function SetBlendWindow( hWnd : HWND; AlphaBlend : Byte) : LongBool
5159:   Function EditWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM) : LRESULT
5160:   Procedure SetEditWndProc( hWnd : HWND; ptr : TObject)
5161:   Function GetTextFromFile( Filename : String) : string
5162:   Function IsTopMost( hWnd : HWND) : Bool // 'LWA_ALPHA','LongWord').SetUInt( $00000002);
5163:   Function msStrToIntDef( const s : String; const i : Integer) : Integer
5164:   Function msStrToInt( s : String) : Integer
5165:   Function GetItemText( hDlg : THandle; ID : DWORD) : String
5166: end;
5167:
5168: procedure SIRegister_ESBMaths2(CL: TPSPascalCompiler);
5169: begin
5170:   //TDynFloatArray', 'array of Extended
5171:   TDynLWordArray', 'array of LongWord
5172:   TDynLIntArray', 'array of LongInt
5173:   TDynFloatMatrix', 'array of TDynFloatArray
5174:   TDynLWordMatrix', 'array of TDynLWordArray
5175:   TDynLIntMatrix', 'array of TDynLIntArray
5176:   Function SquareAll( const X : TDynFloatArray) : TDynFloatArray
5177:   Function InverseAll( const X : TDynFloatArray) : TDynFloatArray
5178:   Function LnAll( const X : TDynFloatArray) : TDynFloatArray
5179:   Function Log10All( const X : TDynFloatArray) : TDynFloatArray
5180:   Function LinearTransform( const X : TDynFloatArray; Offset, Scale : Extended) : TDynFloatArray
5181:   Function AddVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5182:   Function SubVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5183:   Function MultVectors( const X, Y : TDynFloatArray) : TDynFloatArray
5184:   Function DotProduct( const X, Y : TDynFloatArray) : Extended
5185:   Function MNorm( const X : TDynFloatArray) : Extended
5186:   Function MatrixIsRectangular( const X : TDynFloatMatrix) : Boolean
5187:   Procedure MatrixDimensions(const X:TDynFloatMatrix;var Rows,Columns:LongWord;var Rectangular:Boolean;
5188:   Function MatrixIsSquare( const X : TDynFloatMatrix) : Boolean
5189:   Function MatricesSameDimensions( const X, Y : TDynFloatMatrix) : Boolean
5190:   Function AddMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5191:   Procedure AddToMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5192:   Function SubtractMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix
5193:   Procedure SubtractFromMatrix( var X : TDynFloatMatrix; const Y : TDynFloatMatrix)
5194:   Function MultiplyMatrixByConst( const X : TDynFloatMatrix; const K : Extended) : TDynFloatMatrix
5195:   Procedure MultiplyMatrixByConst2( var X : TDynFloatMatrix; const K : Extended);
5196:   Function MultiplyMatrices( const X, Y : TDynFloatMatrix) : TDynFloatMatrix;
5197:   Function TransposeMatrix( const X : TDynFloatMatrix) : TDynFloatMatrix;
5198:   Function GrandMean( const X : TDynFloatMatrix; var N : LongWord) : Extended
5199: end;
5200:
5201: procedure SIRegister_ESBMaths(CL: TPSPascalCompiler);
5202: begin
5203:   'ESBMinSingle', 'Single').setExtended( 1.5e-45);
5204:   'ESBMaxSingle', 'Single').setExtended( 3.4e+38);
5205:   'ESBMinDouble', 'Double').setExtended( 5.0e-324);
5206:   'ESBMaxDouble', 'Double').setExtended( 1.7e+308);
5207:   'ESBMinExtended', 'Extended').setExtended( 3.6e-4951);
5208:   'ESBMaxExtended', 'Extended').setExtended( 1.1e+4932);
5209:   'ESBMinCurrency', 'Currency').SetExtended( - 922337203685477.5807);
5210:   'ESBMaxCurrency', 'Currency').SetExtended( 922337203685477.5807);
5211:   'ESBSqrt2', 'Extended').setExtended( 1.4142135623730950488);
5212:   'ESBSqrt3', 'Extended').setExtended( 1.7320508075688772935);
5213:   'ESBSqrt5', 'Extended').setExtended( 2.2360679774997896964);
5214:   'ESBSqrt10', 'Extended').setExtended( 3.1622776601683793320);
5215:   'ESBSqrtPi', 'Extended').setExtended( 1.77245385090551602729);
5216:   'ESBCbrt2', 'Extended').setExtended( 1.2599210498948731648);
5217:   'ESBCbrt3', 'Extended').setExtended( 1.4422495703074083823);
5218:   'ESBCbrt10', 'Extended').setExtended( 2.1544346900318837219);
5219:   'ESBCbrt100', 'Extended').setExtended( 4.6415888336127788924);
5220:   'ESBCbrtPi', 'Extended').setExtended( 1.4645918875615232630);
5221:   'ESBInvSqrt2', 'Extended').setExtended( 0.70710678118654752440);
5222:   'ESBInvSqrt3', 'Extended').setExtended( 0.57735026918962576451);
5223:   'ESBInvSqrt5', 'Extended').setExtended( 0.44721359549995793928);
5224:   'ESBInvSqrtPi', 'Extended').setExtended( 0.56418958354775628695);
5225:   'ESBInvCbrtPi', 'Extended').setExtended( 0.68278406325529568147);
5226:   'ESBe', 'Extended').setExtended( 2.7182818284590452354);

```



```

5227: 'ESBe2','Extended').setExtended( 7.3890560989306502272);
5228: 'ESBePi','Extended').setExtended( 23.140692632779269006);
5229: 'ESBePiOn2','Extended').setExtended( 4.8104773809653516555);
5230: 'ESBePiOn4','Extended').setExtended( 2.1932800507380154566);
5231: 'ESBLn2','Extended').setExtended( 0.69314718055994530942);
5232: 'ESBLn10','Extended').setExtended( 2.30258509299404568402);
5233: 'ESBLnP1','Extended').setExtended( 1.14472988584940017414);
5234: 'ESBLog10Base2','Extended').setExtended( 3.3219280948873623478);
5235: 'ESBLog2Base10','Extended').setExtended( 0.30102999566398119521);
5236: 'ESBLog3Base10','Extended').setExtended( 0.47712125471966243730);
5237: 'ESBLogPiBase10','Extended').setExtended( 0.4971498726941339);
5238: 'ESBLogEBase10','Extended').setExtended( 0.43429448190325182765);
5239: 'ESBPi','Extended').setExtended( 3.1415926535897932385);
5240: 'ESBInvPi','Extended').setExtended( 3.1830988618379067154e-1);
5241: 'ESBTwoPi','Extended').setExtended( 6.2831853071795864769);
5242: 'ESBThreePi','Extended').setExtended( 9.4247779607693797153);
5243: 'ESBPi2','Extended').setExtended( 9.8696044010893586188);
5244: 'ESBPiToE','Extended').setExtended( 22.459157718361045473);
5245: 'ESBPiOn2','Extended').setExtended( 1.5707963267948966192);
5246: 'ESBPiOn3','Extended').setExtended( 1.0471975511965977462);
5247: 'ESBPiOn4','Extended').setExtended( 0.7853981633974483096);
5248: 'ESBThreePiOn2','Extended').setExtended( 4.7123889803846898577);
5249: 'ESBFourPiOn3','Extended').setExtended( 4.1887902047863909846);
5250: 'ESBTwoToPower63','Extended').setExtended( 9223372036854775808.0);
5251: 'ESBOneRadian','Extended').setExtended( 57.295779513082320877);
5252: 'ESBOneDegree','Extended').setExtended( 1.7453292519943295769E-2);
5253: 'ESBOneMinute','Extended').setExtended( 2.9088820866572159615E-4);
5254: 'ESBOneSecond','Extended').setExtended( 4.8481368110953599359E-6);
5255: 'ESBGamma','Extended').setExtended( 0.57721566490153286061);
5256: 'ESBLnRt2Pi','Extended').setExtended( 9.189385332046727E-1);
5257: //LongWord', 'Cardinal
5258: TBitList', 'Word
5259: Function UMul( const Num1, Num2 : LongWord) : LongWord
5260: Function UMulDiv2p32( const Num1, Num2 : LongWord) : LongWord
5261: Function UMulDiv( const Num1, Num2, Divisor : LongWord) : LongWord
5262: Function UMulMod( const Num1, Num2, Modulus : LongWord) : LongWord
5263: Function SameFloat( const X1, X2 : Extended) : Boolean
5264: Function FloatIsZero( const X : Extended) : Boolean
5265: Function FloatIsPositive( const X : Extended) : Boolean
5266: Function FloatIsNegative( const X : Extended) : Boolean
5267: Procedure IncLim( var B : Byte; const Limit : Byte)
5268: Procedure IncLimSI( var B : ShortInt; const Limit : ShortInt)
5269: Procedure IncLimW( var B : Word; const Limit : Word)
5270: Procedure IncLimI( var B : Integer; const Limit : Integer)
5271: Procedure IncLimL( var B : LongInt; const Limit : LongInt)
5272: Procedure DecLim( var B : Byte; const Limit : Byte)
5273: Procedure DecLimSI( var B : ShortInt; const Limit : ShortInt)
5274: Procedure DecLimW( var B : Word; const Limit : Word)
5275: Procedure DecLimI( var B : Integer; const Limit : Integer)
5276: Procedure DecLimL( var B : LongInt; const Limit : LongInt)
5277: Function MaxB( const B1, B2 : Byte) : Byte
5278: Function MinB( const B1, B2 : Byte) : Byte
5279: Function MaxSI( const B1, B2 : ShortInt) : ShortInt
5280: Function MinSI( const B1, B2 : ShortInt) : ShortInt
5281: Function MaxW( const B1, B2 : Word) : Word
5282: Function MinW( const B1, B2 : Word) : Word
5283: Function esbMaxI( const B1, B2 : Integer) : Integer
5284: Function esbMinI( const B1, B2 : Integer) : Integer
5285: Function MaxL( const B1, B2 : LongInt) : LongInt
5286: Function MinL( const B1, B2 : LongInt) : LongInt
5287: Procedure SwapB( var B1, B2 : Byte)
5288: Procedure SwapSI( var B1, B2 : ShortInt)
5289: Procedure SwapW( var B1, B2 : Word)
5290: Procedure SwapI( var B1, B2 : SmallInt)
5291: Procedure SwapL( var B1, B2 : LongInt)
5292: Procedure SwapI32( var B1, B2 : Integer)
5293: Procedure SwapC( var B1, B2 : LongWord)
5294: Procedure SwapInt64( var X, Y : Int64)
5295: Function esbSign( const B : LongInt) : ShortInt
5296: Function Max4Word( const X1, X2, X3, X4 : Word) : Word
5297: Function Min4Word( const X1, X2, X3, X4 : Word) : Word
5298: Function Max3Word( const X1, X2, X3 : Word) : Word
5299: Function Min3Word( const X1, X2, X3 : Word) : Word
5300: Function MaxBArray( const B : array of Byte) : Byte
5301: Function MaxWArray( const B : array of Word) : Word
5302: Function MaxSIArray( const B : array of ShortInt) : ShortInt
5303: Function MaxiArray( const B : array of Integer) : Integer
5304: Function MaxLArray( const B : array of LongInt) : LongInt
5305: Function MinBArray( const B : array of Byte) : Byte
5306: Function MinWArray( const B : array of Word) : Word
5307: Function MinSIArray( const B : array of ShortInt) : ShortInt
5308: Function MiniArray( const B : array of Integer) : Integer
5309: Function MinLArray( const B : array of LongInt) : LongInt
5310: Function SumBArray( const B : array of Byte) : Byte
5311: Function SumBArray2( const B : array of Byte) : Word
5312: Function SumSIArray( const B : array of ShortInt) : ShortInt
5313: Function SumSIArray2( const B : array of ShortInt) : Integer
5314: Function SumWArray( const B : array of Word) : Word
5315: Function SumWArray2( const B : array of Word) : LongInt

```

```

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

```

```

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

```

```

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

```



```

5570: begin
5571:   TConstArray', 'array of TVarRec
5572:   Function CopyVarRec( const Item : TVarRec ) : TVarRec
5573:   Function CreateConstArray( const Elements : array of const ) : TConstArray
5574:   Procedure FinalizeVarRec( var Item : TVarRec )
5575:   Procedure FinalizeConstArray( var Arr : TConstArray )
5576: end;
5577:
5578: procedure SIRegister_StStrS(CL: TPSPascalCompiler);
5579: begin
5580:   Function HexBS( B : Byte ) : ShortString
5581:   Function HexWS( W : Word ) : ShortString
5582:   Function HexLS( L : LongInt ) : ShortString
5583:   Function HexPtrS( P : Pointer ) : ShortString
5584:   Function BinaryBS( B : Byte ) : ShortString
5585:   Function BinaryWS( W : Word ) : ShortString
5586:   Function BinaryLS( L : LongInt ) : ShortString
5587:   Function OctalBS( B : Byte ) : ShortString
5588:   Function OctalWS( W : Word ) : ShortString
5589:   Function OctalLS( L : LongInt ) : ShortString
5590:   Function Str2Int16S( const S : ShortString; var I : SmallInt ) : Boolean
5591:   Function Str2Words( const S : ShortString; var I : Word ) : Boolean
5592:   Function Str2LongS( const S : ShortString; var I : LongInt ) : Boolean
5593:   Function Str2RealS( const S : ShortString; var R : Double ) : Boolean
5594:   Function Str2RealS( const S : ShortString; var R : Real ) : Boolean
5595:   Function Str2ExtS( const S : ShortString; var R : Extended ) : Boolean
5596:   Function Long2StrS( L : LongInt ) : ShortString
5597:   Function Real2StrS( R : Double; Width : Byte; Places : ShortInt ) : ShortString
5598:   Function Ext2StrS( R : Extended; Width : Byte; Places : ShortInt ) : ShortString
5599:   Function ValPrepS( const S : ShortString ) : ShortString
5600:   Function CharStrS( C : AnsiChar; Len : Cardinal ) : ShortString
5601:   Function PadChS( const S : ShortString; C : AnsiChar; Len : Cardinal ) : ShortString
5602:   Function PadS( const S : ShortString; Len : Cardinal ) : ShortString
5603:   Function LeftPadChS( const S : ShortString; C : AnsiChar; Len : Cardinal ) : ShortString
5604:   Function LeftPadS( const S : ShortString; Len : Cardinal ) : ShortString
5605:   Function TrimLeadS( const S : ShortString ) : ShortString
5606:   Function TrimTrails( const S : ShortString ) : ShortString
5607:   Function TrimS( const S : ShortString ) : ShortString
5608:   Function TrimSpacesS( const S : ShortString ) : ShortString
5609:   Function CenterChS( const S : ShortString; C : AnsiChar; Len : Cardinal ) : ShortString
5610:   Function CenterS( const S : ShortString; Len : Cardinal ) : ShortString
5611:   Function EntabS( const S : ShortString; TabSize : Byte ) : ShortString
5612:   Function DetabS( const S : ShortString; TabSize : Byte ) : ShortString
5613:   Function ScrambleS( const S, Key : ShortString ) : ShortString
5614:   Function SubstituteS( const S, FromStr, ToStr : ShortString ) : ShortString
5615:   Function FilterS( const S, Filters : ShortString ) : ShortString
5616:   Function CharExistsS( const S : ShortString; C : AnsiChar ) : Boolean
5617:   Function CharCountS( const S : ShortString; C : AnsiChar ) : Byte
5618:   Function WordCountS( const S, WordDelims : ShortString ) : Cardinal
5619:   Function WordPositionS( N : Cardinal; const S, WordDelims : ShortString; var Pos : Cardinal ) : Boolean
5620:   Function ExtractWords( N : Cardinal; const S, WordDelims : ShortString ) : ShortString
5621:   Function AsciiCountS( const S, WordDelims : ShortString; Quote : AnsiChar ) : Cardinal
5622:   Function AsciiPositionS( N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar;var Pos:Cardinal):Boolean
5623:   Function ExtractAsciiS(N:Cardinal;const S,WordDelims:ShortString;Quote:AnsiChar): ShortString
5624:   Procedure WordWrapS(const InSt: ShortString; var OutSt,Overlap: ShortString;
Margin:Cardinal;PadToMargin:Boolean)
5625:   Function CompStringS( const S1, S2 : ShortString ) : Integer
5626:   Function CompUCStringS( const S1, S2 : ShortString ) : Integer
5627:   Function SoundexS( const S : ShortString ) : ShortString
5628:   Function MakeLetterSetS( const S : ShortString ) : Longint
5629:   Procedure BMMakeTableS( const MatchString : ShortString; var BT : BTable )
5630:   Function BMSearchS(var Buffer,BufLength:Card;var BT:BTable;const MatchString:ShortString;var
Pos:Cardinal):Bool;
5631:   Function BMSearchUCS(var Buffer,BufLength:Card;var BT:BTable;const MatchString:ShortString;var
Pos:Cardinal):Bool;
5632:   Function DefaultExtensionS( const Name, Ext : ShortString ) : ShortString
5633:   Function ForceExtensionS( const Name, Ext : ShortString ) : ShortString
5634:   Function JustFilenameS( const PathName : ShortString ) : ShortString
5635:   Function JustNameS( const PathName : ShortString ) : ShortString
5636:   Function JustExtensionS( const Name : ShortString ) : ShortString
5637:   Function JustPathNameS( const PathName : ShortString ) : ShortString
5638:   Function AddBackSlashS( const DirName : ShortString ) : ShortString
5639:   Function CleanPathNameS( const PathName : ShortString ) : ShortString
5640:   Function HasExtensionS( const Name : ShortString; var DotPos : Cardinal ) : Boolean
5641:   Function CommaizeS( L : LongInt ) : ShortString
5642:   Function CommaizeChS( L : LongInt; Ch : AnsiChar ) : ShortString
5643:   Function FloatFormS(const Mask:ShortString;R:TstFloat;const LtCurr,RtCurr:ShortString;Sep,
DecPt:Char): ShortString;
5644:   Function LongIntFormS(const Mask:ShortString;L:LongInt;const LtCurr,
RtCurr:ShortString;Sep:AnsiChar):ShortString;
5645:   Function StrChPosS( const P : ShortString; C : AnsiChar; var Pos : Cardinal ) : Boolean
5646:   Function StrStPosS( const P, S : ShortString; var Pos : Cardinal ) : Boolean
5647:   Function StrStCopyS( const S : ShortString; Pos, Count : Cardinal ) : ShortString
5648:   Function StrChInsertS( const S : ShortString; C : AnsiChar; Pos : Cardinal ) : ShortString
5649:   Function StrStInsertS( const S1, S2 : ShortString; Pos : Cardinal ) : ShortString
5650:   Function StrChDeleteS( const S : ShortString; Pos : Cardinal ) : ShortString
5651:   Function StrStDeleteS( const S : ShortString; Pos, Count : Cardinal ) : ShortString
5652:   Function ContainsOnlyS( const S, Chars : ShortString; var BadPos : Cardinal ) : Boolean
5653:   Function ContainsOtherThanS( const S, Chars : ShortString; var BadPos : Cardinal ) : Boolean

```

```

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

```

```

5729: Function DiscountRate(Settlement,Maturity:TStDate; Price,Redemption:Extended;Basis:TStBasis): Extended;
5730: Function DollarToDecimal( FracDollar : Extended; Fraction : Integer) : Extended
5731: Function DollarToDecimalText( DecDollar : Extended) : string
5732: Function DollarToFraction( DecDollar : Extended; Fraction : Integer) : Extended
5733: Function DollarToFractionStr( FracDollar : Extended; Fraction : Integer) : string
5734: Function EffectiveInterestRate( NominalRate : Extended; Frequency : TStFrequency) : Extended
5735: Function FutureValueS(Rate:Extended;NPeriods:Integer;Pmt,
PV:Extended;Freq:TStFrequency;Timing:TStPaymentTime):Extended;
5736: Function FutureValueSchedule( Principal : Extended; const Schedule : array of Double) : Extended
5737: Function FutureValueSchedule16( Principal : Extended; const Schedule, NRates : Integer) : Extended
5738: Function InterestRateS(NPeriods:Integer;Pmt,PV,
FV:Extended;Freq:TStFrequency;Timing:TStPaymentTime;Guess:Extended):Extended;
5739: Function InternalRateOfReturn( const Values : array of Double; Guess : Extended) : Extended
5740: Function InternalRateOfReturn16( const Values, NValues : Integer; Guess : Extended) : Extended
5741: Function IsCardValid( const S : string) : Boolean
5742: Function ModifiedDuration(Settlement,Maturity:TStDate;Rate,
Yield:Extended;Freq:TStFrequency;Basis:TStBasis):Extended;
5743: Function ModifiedIRR(const Values : array of Double; FinanceRate,ReinvestRate: Extended) : Extended
5744: Function ModifiedIRR16(const Values, NValues:Integer; FinanceRate,ReinvestRate: Extended) : Extended
5745: Function NetPresentValueS( Rate : Extended; const Values : array of Double) : Extended
5746: Function NetPresentValue16( Rate : Extended; const Values, NValues : Integer) : Extended
5747: Function NominalInterestRate( EffectRate : Extended; Frequency : TStFrequency) : Extended
5748: Function NonperiodicIRR(const Values:array of Double;const Dates:array of TStDate;Guess:Extended):Extended;
5749: Function NonperiodicNPV(Rate:Extended;const Values : array of Double;const Dates:array of TStDate):Extended;
5750: Function Payment( Rate : Extended; NPeriods : Integer; PV, FV : Extended; Frequency : TStFrequency; Timing
: TStPaymentTime): Extended
5751: Function Periods(Rate:Extended;Pmt,PV,FV:Extended;Frequency:TStFrequency;Timing:TStPaymentTime):Integer;
5752: Function PresentValueS( Rate: Extended; NPeriods: Integer; Pmt, FV: Extended; Frequency: TStFrequency;
Timing: TStPaymentTime): Extended
5753: Function ReceivedAtMaturity(Settlement,Maturity:TStDate;Invest,Discount:Extended;Basis:TStBasis):Extended;
5754: Function RoundToDecimal( Value : Extended; Places : Integer; Bankers : Boolean) : Extended
5755: Function TBillEquivYield( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5756: Function TBillPrice( Settlement, Maturity : TStDate; Discount : Extended) : Extended
5757: Function TBillYield( Settlement, Maturity : TStDate; Price : Extended) : Extended
5758: Function VariableDecliningBalance( Cost, Salvage : Extended; Life : Integer; StartPeriod, EndPeriod,
Factor : Extended; NoSwitch : boolean) : Extended
5759: Function YieldDiscounted(Settlement,Maturity:TStDate;Price,Redemption:Extended;Basis:TStBasis):Extended;
5760: Function YieldPeriodic(Settlement,Maturity:TStDate;Rate,Price,
Redemption:Extended;Freq:TStFrequency;Basis:TStBasis): Extended
5761: Function YieldMaturity(Issue,Settlement,Maturity:TStDate;Rate,Price:Extended;Basis:TStBasis):Extended;
5762:
5763: *****unit uPSI_StAstroP;
5764: Procedure PlanetsPos( JD : Double; var PA : TStPlanetsArray)
5765: *****unit unit uPSI_StStat; Statistic Package of SysTools*****
5766: Function AveDev( const Data : array of Double) : Double
5767: Function AveDev16( const Data, NData : Integer) : Double
5768: Function Confidence( Alpha, StandardDev : Double; Size : LongInt) : Double
5769: Function Correlation( const Data1, Data2 : array of Double) : Double
5770: Function Correlation16( const Data1, Data2, NData : Integer) : Double
5771: Function Covariance( const Data1, Data2 : array of Double) : Double
5772: Function Covariance16( const Data1, Data2, NData : Integer) : Double
5773: Function DevSq( const Data : array of Double) : Double
5774: Function DevSq16( const Data, NData : Integer) : Double
5775: Procedure Frequency(const Data:array of Double;const Bins:array of Double;var Counts : array of LongInt);
5776: Procedure Frequency16( const Data, NData : Integer; const Bins, NBins : Integer; var Counts)
5777: Function GeometricMeanS( const Data : array of Double) : Double
5778: Function GeometricMean16( const Data, NData : Integer) : Double
5779: Function HarmonicMeanS( const Data : array of Double) : Double
5780: Function HarmonicMean16( const Data, NData : Integer) : Double
5781: Function Largest( const Data : array of Double; K : Integer) : Double
5782: Function Largest16( const Data, NData : Integer; K : Integer) : Double
5783: Function MedianS( const Data : array of Double) : Double
5784: Function Median16( const Data, NData : Integer) : Double
5785: Function Mode( const Data : array of Double) : Double
5786: Function Model16( const Data, NData : Integer) : Double
5787: Function Percentile( const Data : array of Double; K : Double) : Double
5788: Function Percentile16( const Data, NData : Integer; K : Double) : Double
5789: Function PercentRank( const Data : array of Double; X : Double) : Double
5790: Function PercentRank16( const Data, NData : Integer; X : Double) : Double
5791: Function Permutations( Number, NumberChosen : Integer) : Extended
5792: Function Combinations( Number, NumberChosen : Integer) : Extended
5793: Function FactorialS( N : Integer) : Extended
5794: Function Rank( Number : Double; const Data : array of Double; Ascending : Boolean) : Integer
5795: Function Rank16( Number : Double; const Data, NData : Integer; Ascending : Boolean) : Integer
5796: Function Smallest( const Data : array of Double; K : Integer) : Double
5797: Function Smallest16( const Data, NData : Integer; K : Integer) : Double
5798: Function TrimMean( const Data : array of Double; Percent : Double) : Double
5799: Function TrimMean16( const Data, NData : Integer; Percent : Double) : Double
5800: AddTypeS('TStLinEst', 'record B0 : Double; B1 : double; seB0 : double; seB'
5801: + '1 : Double; R2 : Double; sigma :Double; SSr: double; SSe: Double; F0 : Double; df : Integer;end
5802: Procedure LinEst(const KnownY:array of Double;const KnownX:array of Double;var
LF:TStLinEst;ErrorStats:Bool;
5803: Procedure LogEst(const KnownY:array of Double;const KnownX:array of Double;var
LF:TStLinEst;ErrorStats:Bool;
5804: Function Forecast(X: Double; const KnownY: array of Double; const KnownX: array of Double) : Double
5805: Function ForecastExponential(X:Double;const KnownY:array of Double;const KnownX:array of Double):Double;
5806: Function Intercept( const KnownY : array of Double; const KnownX : array of Double) : Double
5807: Function RSquared( const KnownY : array of Double; const KnownX : array of Double) : Double
5808: Function Slope( const KnownY : array of Double; const KnownX : array of Double) : Double

```

```

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

```



```

5898: Procedure DateTimeDiff(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;var Days:LongInt;var Secs:LongInt
5899: Procedure IncDateTime(const DT1:TStDateTimeRec;var DT2:TStDateTimeRec;Days:Integer;Secs:LongInt)
5900: Function DateTimeToStDate( DT : TDateTime ) : TStDate
5901: Function DateTimeToStTime( DT : TDateTime ) : TStTime
5902: Function StDateToDateTime( D : TStDate ) : TDateTime
5903: Function StTimeToDateTime( T : TStTime ) : TDateTime
5904: Function Convert2ByteDate( TwoByteDate : Word ) : TStDate
5905: Function Convert4ByteDate( FourByteDate : TStDate ) : Word
5906:
5907: Procedure SIRegister_StDateSt(CL: TPSPascalCompiler);
5908: Function DateStringHMStoAstJD( const Picture, DS : string; H, M, S, Epoch : integer ) : Double
5909: Function MonthToString( const Month : Integer ) : string
5910: Function DateStringToStDate( const Picture, S : string; Epoch : Integer ) : TStDate
5911: Function DateStringToDMY(const Picture,S:string; Epoch:Integer; var D, M, Y : Integer):Boolean
5912: Function StDateToDateString( const Picture : string; const Julian : TStDate; Pack : Boolean):string
5913: Function DayOfWeekToString( const WeekDay : TStDayType ) : string
5914: Function DMYtoDateString(const Picture:string;Day,Month,Year,Epoch:Integer;Pack:Boolean): string);
5915: Function CurrentDateString( const Picture : string; Pack : Boolean ) : string
5916: Function CurrentTimeString( const Picture : string; Pack : Boolean ) : string
5917: Function TimeStringToHMS( const Picture, St : string; var H, M, S : Integer ) : Boolean
5918: Function TimeStringToStTime( const Picture, S : string ) : TStTime
5919: Function StTimeToAmPmString( const Picture : string; const T : TStTime; Pack : Boolean ) : string
5920: Function StTimeToTimeString( const Picture : string; const T : TStTime; Pack : Boolean ) : string
5921: Function DateStringIsBlank( const Picture, S : string ) : Boolean
5922: Function InternationalDate( ForceCentury : Boolean ) : string
5923: Function InternationalLongDate( ShortNames : Boolean; ExcludeDOW : Boolean ) : string
5924: Function InternationalTime( ShowSeconds : Boolean ) : string
5925: Procedure ResetInternationalInfo
5926:
5927: procedure SIRegister_StBase(CL: TPSPascalCompiler);
5928: Function DestroyNode( Container : TStContainer; Node : TStNode; OtherData : Pointer ) : Boolean
5929: Function AnsiUpperCaseShort32( const S : string ) : string
5930: Function AnsiCompareTextShort32( const S1, S2 : string ) : Integer
5931: Function AnsiCompareStrShort32( const S1, S2 : string ) : Integer
5932: Function HugeDecompressRLE( const InBuffer, InLen : Longint; var OutBuffer ) : Longint
5933: Function HugeDecompressRLE( const InBuffer, InLen : Longint; var OutBuffer,OutLen:LongInt): Longint
5934: Procedure HugeFillChar( var Dest, Count : Longint; Value : Byte)
5935: Procedure HugeFillStruc( var Dest, Count : Longint; const Value, ValSize : Cardinal)
5936: Function Upcase( C : AnsiChar ) : AnsiChar
5937: Function LcCase( C : AnsiChar ) : AnsiChar
5938: Function CompareLetterSets( Set1, Set2 : LongInt ) : Cardinal
5939: Function CompStruct( const S1, S2, Size : Cardinal ) : Integer
5940: Function Search(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardinal):Bool;
5941: Function StSearch(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
5942: Function SearchUC(const Buffer,BufLength:Cardinal;const Match,MatLength:Cardinal;var Pos:Cardi):Boolean
5943: Function IsOrInheritsFrom( Root, Candidate : TClass ) : boolean
5944: Procedure RaiseContainerError( Code : longint)
5945: Procedure RaiseContainerErrorFmt( Code : Longint; Data : array of const)
5946: Function ProductOverflow( A, B : LongInt ) : Boolean
5947: Function StNewStr( S : string ) : PShortString
5948: Procedure StDisposeStr( PS : PShortString)
5949: Procedure VallongInt( S : ShortString; var LI : Longint; var ErrorCode : integer)
5950: Procedure ValSmallint( const S : ShortString; var SI : smallint; var ErrorCode : integer)
5951: Procedure ValWord( const S : ShortString; var Wd : word; var ErrorCode : integer)
5952: Procedure RaiseStError( ExceptionClass : EStExceptionClass; Code : LongInt)
5953: Procedure RaiseStWin32Error( ExceptionClass : EStExceptionClass; Code : LongInt)
5954: Procedure RaiseStWin32ErrorEx( ExceptionClass : EStExceptionClass; Code : LongInt; Info : string)
5955:
5956: procedure SIRegister_usvdl(CL: TPSPascalCompiler);
5957: begin
5958: Procedure SV_Decom( A : TMatrix; Lb, Ub1, Ub2 : Integer; S : TVector; V : TMatrix)
5959: Procedure SV_SetZero( S : TVector; Lb, Ub : Integer; Tol : Float)
5960: Procedure SV_Solve(U:TMatrix; S:TVector;V:TMatrix;B:TVector;Lb,Ub1,Ub2:Integer;X:TVector);
5961: Procedure SV_Approx( U : TMatrix; S : TVector; V : TMatrix; Lb, Ub1, Ub2 : Integer; A : TMatrix)
5962: Procedure RKF45(F:TDiffEqs;Negn:Int;Y,Yp:TVector;var T:Float;Tout,RelErr,AbsErr:Float;var Flag:Int;
5963: end;
5964:
5965: //*****unit unit ; StMath Package of SysTools*****
5966: Function IntPowerS( Base : Extended; Exponent : Integer ) : Extended
5967: Function PowerS( Base, Exponent : Extended ) : Extended
5968: Function StInvCos( X : Double ) : Double
5969: Function StInvSin( Y : Double ) : Double
5970: Function StInvTan2( X, Y : Double ) : Double
5971: Function StTan( A : Double ) : Double
5972: Procedure DumpException; //unit StExpEng;
5973: Function HexifyBlock( var Buffer, BufferSize : Integer ) : string
5974:
5975: //*****unit unit ; StCRC Package of SysTools*****
5976: Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt ) : LongInt
5977: Function Adler32OfStream( Stream : TStream; CurCrc : LongInt ) : LongInt
5978: Function Adler32OfFile( FileName : AnsiString ) : LongInt
5979: Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal ) : Cardinal
5980: Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal ) : Cardinal
5981: Function Crc16OfFile( FileName : AnsiString ) : Cardinal
5982: Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt ) : LongInt
5983: Function Crc32OfStream( Stream : TStream; CurCrc : LongInt ) : LongInt
5984: Function Crc32OfFile( FileName : AnsiString ) : LongInt
5985: Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal ) : Cardinal
5986: Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal ) : Cardinal

```

```

5987: Function InternetSumOfFile( FileName : AnsiString) : Cardinal
5988: Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
5989: Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
5990: Function Kermit16OfFile( FileName : AnsiString) : Cardinal
5991:
5992: *****unit unit ; StBCD Package of SysTools*****
5993: Function AddBcd( const B1, B2 : TbcdS) : TbcdS
5994: Function SubBcd( const B1, B2 : TbcdS) : TbcdS
5995: Function MulBcd( const B1, B2 : TbcdS) : TbcdS
5996: Function DivBcd( const B1, B2 : TbcdS) : TbcdS
5997: Function ModBcd( const B1, B2 : TbcdS) : TbcdS
5998: Function NegBcd( const B : TbcdS) : TbcdS
5999: Function AbsBcd( const B : TbcdS) : TbcdS
6000: Function FracBcd( const B : TbcdS) : TbcdS
6001: Function IntBcd( const B : TbcdS) : TbcdS
6002: Function RoundDigitsBcd( const B : TbcdS; Digits : Cardinal) : TbcdS
6003: Function RoundPlacesBcd( const B : TbcdS; Places : Cardinal) : TbcdS
6004: Function ValBcd( const S : string) : TbcdS
6005: Function LongBcd( L : LongInt) : TbcdS
6006: Function ExtBcd( E : Extended) : TbcdS
6007: Function ExpBcd( const B : TbcdS) : TbcdS
6008: Function LnBcd( const B : TbcdS) : TbcdS
6009: Function IntPowBcd( const B : TbcdS; E : LongInt) : TbcdS
6010: Function PowBcd( const B, E : TbcdS) : TbcdS
6011: Function SqrtBcd( const B : TbcdS) : TbcdS
6012: Function CmpBcd( const B1, B2 : TbcdS) : Integer
6013: Function EqDigitsBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6014: Function EqPlacesBcd( const B1, B2 : TbcdS; Digits : Cardinal) : Boolean
6015: Function IsIntBcd( const B : TbcdS) : Boolean
6016: Function TruncBcd( const B : TbcdS) : LongInt
6017: Function BcdExt( const B : TbcdS) : Extended
6018: Function RoundBcd( const B : TbcdS) : LongInt
6019: Function StrBcd( const B : TbcdS; Width, Places : Cardinal) : string
6020: Function StrExpBcd( const B : TbcdS; Width : Cardinal) : string
6021: Function FormatBcd( const Format : string; const B : TbcdS) : string
6022: Function StrGeneralBcd( const B : TbcdS) : string
6023: Function FloatFormBcd(const Mask:string;B:TbcdS;const LtCurr,RtCurr:string;Sep,DecPt:AnsiChar):string
6024: Procedure ConvertBcd( const SrcB, SrcSize : Byte; var DestB, DestSize : Byte)
6025:
6026: *****unit unit ; StTxtDat; TStTextDataRecordSet Package of SysTools*****
6027: Procedure StParseLine( const Data : AnsiString; Schema : TStTextDataSchema; Result : TStrings)
6028: Function StFieldTypeToStr( FieldType : TStSchemaFieldType) : AnsiString
6029: Function StStrToFieldType( const S : AnsiString) : TStSchemaFieldType
6030: Function StDeEscape( const EscStr : AnsiString) : Char
6031: Function StDoEscape( Delim : Char) : AnsiString
6032: Function StTrimTrailingChars( const S : AnsiString; Trailer : Char) : AnsiString
6033: Function AnsiHashText( const S : string; Size : Integer) : Integer
6034: Function AnsiHashStr( const S : string; Size : Integer) : Integer
6035: Function AnsiELFHashText( const S : string; Size : Integer) : Integer
6036: Function AnsiELFHashStr( const S : string; Size : Integer) : Integer
6037:
6038: *****unit unit ; StNetCon Package of SysTools*****
6039: with AddClassN(FindClass('TStComponent'),'TStNetConnection') do begin
6040:   Constructor Create( AOwner : TComponent)
6041:   Function Connect : DWord
6042:   Function Disconnect : DWord
6043:   RegisterProperty('Password', 'String', iptrw);
6044:   Property('UserName', 'String', iptrw);
6045:   Property('ConnectOptions', 'TStNetConnectOptionsSet', iptrw);
6046:   Property('DisconnectOptions', 'TStNetDisconnectOptionsSet', iptrw);
6047:   Property('LocalDevice', 'String', iptrw);
6048:   Property('ServerName', 'String', iptrw);
6049:   Property('ShareName', 'String', iptrw);
6050:   Property('OnConnect', 'TNotifyEvent', iptrw);
6051:   Property('OnConnectFail', 'TOnConnectFailEvent', iptrw);
6052:   Property('OnConnectCancel', 'TOnConnectCancelEvent', iptrw);
6053:   Property('OnDisconnect', 'TNotifyEvent', iptrw);
6054:   Property('OnDisconnectFail', 'TOnDisconnectFailEvent', iptrw);
6055:   Property('OnDisconnectCancel', 'TOnDisconnectCancelEvent', iptrw);
6056: end;
6057: *****Thread Functions Context of Win API --- more objects in SyncObjs.pas
6058: / /153 unit uPSI_SyncObjs, unit uPSIParallelJobs;
6059: Procedure InitializeCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6060: Procedure EnterCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6061: Procedure LeaveCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6062: Function InitializeCriticalSectionAndSpinCount(var
  lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):BOOL;
6063: Function SetCriticalSectionSpinCount(var lpCriticalSection:TRTLCriticalSection;dwSpinCount:DWORD):DWORD;
6064: Function TryEnterCriticalSection( var lpCriticalSection : TRTLCriticalSection) : BOOL
6065: Procedure DeleteCriticalSection( var lpCriticalSection : TRTLCriticalSection)
6066: Function GetThreadContext( hThread : THandle; var lpContext : TContext) : BOOL
6067: Function SetThreadContext( hThread : THandle; const lpContext : TContext) : BOOL
6068: Function SuspendThread( hThread : THandle) : DWORD
6069: Function ResumeThread( hThread : THandle) : DWORD
6070: Function CreateThread2(ThreadFunc: TThreadFunction2; thrId: DWord) : THandle
6071: Function GetCurrentThread : THandle
6072: Procedure ExitThread( dwExitCode : DWORD)
6073: Function TerminateThread( hThread : THandle; dwExitCode : DWORD) : BOOL
6074: Function GetExitCodeThread( hThread : THandle; var lpExitCode : DWORD) : BOOL

```

```

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

```

```

6161: Function PathCompactPathEx( pszOut : PChar; pszSrc : PChar; cchMax : UINT; dwFlags:DWORD ) : BOOL
6162: Function PathCommonPrefix( pszFile1, pszFile2 : PChar; achPath : PChar ) : Integer
6163: Function PathFileExists( pszPath : PChar ) : BOOL
6164: Function PathFindExtension( pszPath : PChar ) : PChar
6165: Function PathFindFileName( pszPath : PChar ) : PChar
6166: Function PathFindNextComponent( pszPath : PChar ) : PChar
6167: Function PathFindOnPath( pszPath : PChar; var ppszOtherDirs : PChar ) : BOOL
6168: Function PathGetArgs( pszPath : PChar ) : PChar
6169: Function PathFindSuffixArray( pszPath: PChar; const apszSuffix: PChar; iArraySize: Integer): PChar
6170: Function PathIsLFNFileSpec( lpName : PChar ) : BOOL
6171: Function PathGetCharType( ch : Char ) : UINT
6172: GCT_INVALID, 'LongWord').SetUInt( $0000);
6173: GCT_LFNCHAR, 'LongWord').SetUInt( $0001);
6174: GCT_SHORTCHAR, 'LongWord').SetUInt( $0002);
6175: GCT_WILD, 'LongWord').SetUInt( $0004);
6176: GCT_SEPARATOR, 'LongWord').SetUInt( $0008);
6177: Function PathGetDriveNumber( pszPath : PChar ) : Integer
6178: Function PathIsDirectory( pszPath : PChar ) : BOOL
6179: Function PathIsDirectoryEmpty( pszPath : PChar ) : BOOL
6180: Function PathIsFileSpec( pszPath : PChar ) : BOOL
6181: Function PathIsPrefix( pszPrefix, pszPath : PChar ) : BOOL
6182: Function PathIsRelative( pszPath : PChar ) : BOOL
6183: Function PathIsRoot( pszPath : PChar ) : BOOL
6184: Function PathIsSameRoot( pszPath1, pszPath2 : PChar ) : BOOL
6185: Function PathIsUNC( pszPath : PChar ) : BOOL
6186: Function PathIsNetworkPath( pszPath : PChar ) : BOOL
6187: Function PathIsUNCServer( pszPath : PChar ) : BOOL
6188: Function PathIsUNCServerShare( pszPath : PChar ) : BOOL
6189: Function PathIsContentType( pszPath, pszContentType : PChar ) : BOOL
6190: Function PathIsURL( pszPath : PChar ) : BOOL
6191: Function PathMakePretty( pszPath : PChar ) : BOOL
6192: Function PathMatchSpec( pszFile, pszSpec : PChar ) : BOOL
6193: Function PathParseIconLocation( pszIconFile : PChar ) : Integer
6194: Procedure PathQuoteSpaces( lpsz : PChar)
6195: Function PathRelativePathTo( pszPath:PChar; pszFrom:PChar; dwAttrFrom:DWORD; pszTo:PChar; dwAttrTo:DWORD ):BOOL;
6196: Procedure PathRemoveArgs( pszPath : PChar)
6197: Function PathRemoveBackslash( pszPath : PChar ) : PChar
6198: Procedure PathRemoveBlanks( pszPath : PChar)
6199: Procedure PathRemoveExtension( pszPath : PChar)
6200: Function PathRemoveFileSpec( pszPath : PChar ) : BOOL
6201: Function PathRenameExtension( pszPath : PChar; pszExt : PChar ) : BOOL
6202: Function PathSearchAndQualify( pszPath : PChar; pszBuf : PChar; cchBuf : UINT ) : BOOL
6203: Procedure PathSetDlgItemPath( hDlg : HWND; id : Integer; pszPath : PChar)
6204: Function PathSkipRoot( pszPath : PChar ) : PChar
6205: Procedure PathStripPath( pszPath : PChar)
6206: Function PathStripToRoot( pszPath : PChar ) : BOOL
6207: Procedure PathUnquoteSpaces( lpsz : PChar)
6208: Function PathMakeSystemFolder( pszPath : PChar ) : BOOL
6209: Function PathUnmakeSystemFolder( pszPath : PChar ) : BOOL
6210: Function PathIsSystemFolder( pszPath : PChar; dwAttrb : DWORD ) : BOOL
6211: Procedure PathUndecorate( pszPath : PChar)
6212: Function PathUnExpandEnvStrings( pszPath : PChar; pszBuf : PChar; cchBuf : UINT ) : BOOL
6213: URL_SCHEME_INVALID, 'LongInt').SetInt( - 1);
6214: URL_SCHEME_UNKNOWN, 'LongInt').SetInt( 0);
6215: URL_SCHEME_FTP, 'LongInt').SetInt( 1);
6216: URL_SCHEME_HTTP, 'LongInt').SetInt( 2);
6217: URL_SCHEME_GOPHER, 'LongInt').SetInt( 3);
6218: URL_SCHEME_MAILTO, 'LongInt').SetInt( 4);
6219: URL_SCHEME_NEWS, 'LongInt').SetInt( 5);
6220: URL_SCHEME_NNTP, 'LongInt').SetInt( 6);
6221: URL_SCHEME_TELNET, 'LongInt').SetInt( 7);
6222: URL_SCHEME_WAIS, 'LongInt').SetInt( 8);
6223: URL_SCHEME_FILE, 'LongInt').SetInt( 9);
6224: URL_SCHEME_MK, 'LongInt').SetInt( 10);
6225: URL_SCHEME_HTTPS, 'LongInt').SetInt( 11);
6226: URL_SCHEME_SHELL, 'LongInt').SetInt( 12);
6227: URL_SCHEME_SNEWS, 'LongInt').SetInt( 13);
6228: URL_SCHEME_LOCAL, 'LongInt').SetInt( 14);
6229: URL_SCHEME_JAVASCRIPT, 'LongInt').SetInt( 15);
6230: URL_SCHEME_VBSCRIPT, 'LongInt').SetInt( 16);
6231: URL_SCHEME_ABOUT, 'LongInt').SetInt( 17);
6232: URL_SCHEME_RES, 'LongInt').SetInt( 18);
6233: URL_SCHEME_MAXVALUE, 'LongInt').SetInt( 19);
6234: URL_SCHEME, 'Integer
6235: URL_PART_NONE, 'LongInt').SetInt( 0);
6236: URL_PART_SCHEME, 'LongInt').SetInt( 1);
6237: URL_PART_HOSTNAME, 'LongInt').SetInt( 2);
6238: URL_PART_USERNAME, 'LongInt').SetInt( 3);
6239: URL_PART_PASSWORD, 'LongInt').SetInt( 4);
6240: URL_PART_PORT, 'LongInt').SetInt( 5);
6241: URL_PART_QUERY, 'LongInt').SetInt( 6);
6242: URL_PART, 'DWORD
6243: URLIS_URL, 'LongInt').SetInt( 0);
6244: URLIS_OPAQUE, 'LongInt').SetInt( 1);
6245: URLIS_NOHISTORY, 'LongInt').SetInt( 2);
6246: URLIS_FILEURL, 'LongInt').SetInt( 3);
6247: URLIS_APPLIABLE, 'LongInt').SetInt( 4);
6248: URLIS_DIRECTORY, 'LongInt').SetInt( 5);
6249: URLIS_HASQUERY, 'LongInt').SetInt( 6);

```



```

6250: TurlIs', 'DWORD
6251: URL_UNESCAPE', 'LongWord').SetUInt( $10000000);
6252: URL_ESCAPE_UNSAFE', 'LongWord').SetUInt( $20000000);
6253: URL_PLUGGABLE_PROTOCOL', 'LongWord').SetUInt( $40000000);
6254: URL_WININET_COMPATIBILITY', 'LongWord').SetUInt( DWORD( $80000000 ));
6255: URL_DONT_ESCAPE_EXTRA_INFO', 'LongWord').SetUInt( $02000000);
6256: URL_ESCAPE_SPACES_ONLY', 'LongWord').SetUInt( $04000000);
6257: URL_DONT_SIMPLIFY', 'LongWord').SetUInt( $08000000);
6258: URL_NO_META', 'LongWord').SetUInt( URL_DONT_SIMPLIFY);
6259: URL_UNESCAPE_INPLACE', 'LongWord').SetUInt( $00100000);
6260: URL_CONVERT_IF_DOSPATH', 'LongWord').SetUInt( $00200000);
6261: URL_UNESCAPE_HIGH_ANSI_ONLY', 'LongWord').SetUInt( $00400000);
6262: URL_INTERNAL_PATH', 'LongWord').SetUInt( $00800000);
6263: URL_FILE_USE_PATHURL', 'LongWord').SetUInt( $00010000);
6264: URL_ESCAPE_PERCENT', 'LongWord').SetUInt( $00001000);
6265: URL_ESCAPE_SEGMENT_ONLY', 'LongWord').SetUInt( $00002000);
6266: URL_PARTFLAG_KEEPScheme', 'LongWord').SetUInt( $00000001);
6267: URL_APPLY_DEFAULT', 'LongWord').SetUInt( $00000001);
6268: URL_APPLY_GUESSScheme', 'LongWord').SetUInt( $00000002);
6269: URL_APPLY_GUESSFILE', 'LongWord').SetUInt( $00000004);
6270: URL_APPLY_FORCEAPPLY', 'LongWord').SetUInt( $00000008);
6271: Function UrlCompare( psz1, psz2 : PChar; fIgnoreSlash : BOOL ) : Integer
6272: Function UrlCombine(pszBase,pszRelative:PChar;pszCombin:PChar;out pcchCombin:DWORD;dwFlags:DWORD):HRESULT;
6273: Function UrlCanonicalize(pszUrl:PChar;pszCanonicalized:PChar;pcchCanonic:DWORD;dwFlags:DWORD):HRESULT;
6274: Function UrlIsOpaque( pszURL : PChar ) : BOOL
6275: Function UrlIsNoHistory( pszURL : PChar ) : BOOL
6276: Function UrlIsFileUrl( pszURL : PChar ) : BOOL
6277: Function UrlIs( pszUrl : PChar; UrlIs : TurlIs ) : BOOL
6278: Function UrlGetLocation( psz1 : PChar ) : PChar
6279: Function UrlUnescape( pszUrl, pszUnescaped : PChar;pcchUnescaped:DWORD; dwFlags : DWORD ) : HRESULT
6280: Function UrlEscape(pszUrl: PChar; pszEscaped: PChar; pcchEscaped:DWORD; dwFlags : DWORD ) : HRESULT
6281: Function UrlCreateFromPath(pszPath:PChar; pszUrl:PChar;pcchUrl: DWORD; dwFlags : DWORD ) : HRESULT
6282: Function PathCreateFromUrl(pszUrl:PChar; pszPath:PChar; pcchPath:DWORD; dwFlags : DWORD ) : HRESULT
6283: Function UrlHash( pszUrl : PChar; pbHash : BYTE; cbHash : DWORD ) : HRESULT
6284: Function UrlGetPart(pszIn: PChar; pszOut: PChar; pcchOut: DWORD; dwPart,dwFlags: DWORD ) : HRESULT
6285: Function UrlApplyScheme( pszIn : PChar; pszOut : PChar; pcchOut : DWORD; dwFlags : DWORD ) : HRESULT
6286: Function HashData( pbData : BYTE; cbData : DWORD; pbHash : BYTE; cbHash : DWORD ) : HRESULT
6287: Function UrlEscapeSpaces( pszUrl : PChar; pszEscaped : PChar; pcchEscaped : DWORD ) : HRESULT
6288: Function UrlUnescapeInPlace( pszUrl : PChar; dwFlags : DWORD ) : HRESULT
6289: Function SHDeleteEmptyKey( hKey : HKEY; pszSubKey : PChar ) : DWORD
6290: Function SHDeleteKey( hKey : HKEY; pszSubKey : PChar ) : DWORD
6291: Function SHDeleteValue( hKey : HKEY; pszSubKey, pszValue : PChar ) : DWORD
6292: Function SHEnumKeyEx( hKey : HKEY; dwIndex : DWORD; pszName : PChar; var pcchName : DWORD ) : Longint
6293: Function SHEnumValue( hKey : HKEY; dwIndex : DWORD; pszValueName:PChar; var
pcchValueName:DWORD;pdwType:DWORD; pvData : ___Pointer; pcbData : DWORD ) : Longint
6294: Function SHQueryInfoKey(hKey:HKEY;pcSubKeys,pcchMaxSubKeyLen,pcVal,pcchMaxValueNameLen:DWORD):Longint;
6295: Function SHCopyKey( hkeySrc : HKEY; szSrcSubKey : PChar; hkeyDest : HKEY; fReserved : DWORD ) : DWORD
6296: Function SHRegGetPath(hKey:HKEY; pcszSubKey,pcszValue: PChar; pszPath: PChar; dwFlags: DWORD) : DWORD
6297: Function SHRegSetPath( hKey:HKEY; pcszSubKey, pcszValue, pcszPath : PChar; dwFlags : DWORD): DWORD
6298: SHREGDEL_DEFAULT', 'LongWord').SetUInt( $00000000);
6299: SHREGDEL_HKCU', 'LongWord').SetUInt( $00000001);
6300: SHREGDEL_HKLM', 'LongWord').SetUInt( $00000010);
6301: SHREGDEL_BOTH', 'LongWord').SetUInt( $00000011);
6302: SHREGENUM_DEFAULT', 'LongWord').SetUInt( $00000000);
6303: SHREGENUM_HKCU', 'LongWord').SetUInt( $00000001);
6304: SHREGENUM_HKLM', 'LongWord').SetUInt( $00000010);
6305: SHREGENUM_BOTH', 'LongWord').SetUInt( $00000011);
6306: SHREGSET_HKCU', 'LongWord').SetUInt( $00000001);
6307: SHREGSET_FORCE_HKCU', 'LongWord').SetUInt( $00000002);
6308: SHREGSET_HKLM', 'LongWord').SetUInt( $00000004);
6309: SHREGSET_FORCE_HKLM', 'LongWord').SetUInt( $00000008);
6310: TSHRegDelFlags', 'DWORD
6311: TSHRegEnumFlags', 'DWORD
6312: HUSKEY', 'THandle
6313: ASSOCF_INIT_NOREMAPCLSID', 'LongWord').SetUInt( $00000001);
6314: ASSOCF_INIT_BYEXENAME', 'LongWord').SetUInt( $00000002);
6315: ASSOCF_OPEN_BYEXENAME', 'LongWord').SetUInt( $00000002);
6316: ASSOCF_INIT_DEFAULTTOSTAR', 'LongWord').SetUInt( $00000004);
6317: ASSOCF_INIT_DEFAULTTOFOLDER', 'LongWord').SetUInt( $00000008);
6318: ASSOCF_NOUSERSETTINGS', 'LongWord').SetUInt( $00000010);
6319: ASSOCF_NOTRUNCATE', 'LongWord').SetUInt( $00000020);
6320: ASSOCF_VERIFY', 'LongWord').SetUInt( $00000040);
6321: ASSOCF_REMAPRUNDLL', 'LongWord').SetUInt( $00000080);
6322: ASSOCF_NOFIXUPS', 'LongWord').SetUInt( $00000100);
6323: ASSOCF_IGNOREBASECLASS', 'LongWord').SetUInt( $00000200);
6324: ASSOCF', 'DWORD
6325: ASSOCSTR_COMMAND', 'LongInt').SetInt( 1);
6326: ASSOCSTR_EXECUTABLE', 'LongInt').SetInt( 2);
6327: ASSOCSTR_FRIENDLYDOCNAME', 'LongInt').SetInt( 3);
6328: ASSOCSTR_FRIENDLYAPPNAME', 'LongInt').SetInt( 4);
6329: ASSOCSTR_NOOPEN', 'LongInt').SetInt( 5);
6330: ASSOCSTR_SHELLNEWVALUE', 'LongInt').SetInt( 6);
6331: ASSOCSTR_DDECOMMAND', 'LongInt').SetInt( 7);
6332: ASSOCSTR_DDEIFEXEC', 'LongInt').SetInt( 8);
6333: ASSOCSTR_DDEAPPLICATION', 'LongInt').SetInt( 9);
6334: ASSOCSTR_DDETOPIC', 'LongInt').SetInt( 10);
6335: ASSOCSTR_INFOTIP', 'LongInt').SetInt( 11);
6336: ASSOCSTR_MAX', 'LongInt').SetInt( 12);
6337: ASSOCSTR', 'DWORD

```

```

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

```

```

6427: procedure RIRegister_CPort_Routines(S: TPSExec);
6428: Procedure EnumComPorts( Ports : TStrings)
6429: Procedure ListComPorts( Ports : TStrings)
6430: Procedure ComPorts( Ports : TStrings) //Alias to Arduino
6431: Function GetComPorts: TStringlist;
6432: Function StrToBaudRate( Str : string) : TBaudRate
6433: Function StrToStopBits( Str : string) : TStopBits
6434: Function StrToDataBits( Str : string) : TDataBits
6435: Function StrToParity( Str : string) : TParityBits
6436: Function StrToFlowControl( Str : string) : TFlowControl
6437: Function BaudRateToStr( BaudRate : TBaudRate) : string
6438: Function StopBitsToStr( StopBits : TStopBits) : string
6439: Function DataBitsToStr( DataBits : TDataBits) : string
6440: Function ParityToStr( Parity : TParityBits) : string
6441: Function FlowControlToStr( FlowControl : TFlowControl) : string
6442: Function ComErrorsToStr( Errors : TComErrors) : String
6443:
6444: Function GetMessage( var lpMsg : TMsg; hWnd : HWND; wParamFilterMin, wParamFilterMax : UINT) : BOOL
6445: Function DispatchMessage( const lpMsg : TMsg) : Longint
6446: Function TranslateMessage( const lpMsg : TMsg) : BOOL
6447: Function SetMessageQueue( cMessagesMax : Integer) : BOOL
6448: Function PeekMessage(var lpMsg:TMsg; hWnd:HWND;wParamFilterMin,wParamFilterMax,wRemoveMsg:UINT):BOOL
6449: Function GetMessagePos : DWORD
6450: Function GetMessageTime : Longint
6451: Function GetMessageExtraInfo : Longint
6452: Function GetSpecialFolderPath( const FolderName : string; CanCreate : Boolean) : string
6453: Procedure JAddToRecentDocs( const Filename : string)
6454: Procedure ClearRecentDocs
6455: Function ExtractIconFromFile( FileName : string; Index : Integer) : HICON
6456: Function CreateShellLink( const AppName, Desc : string; Dest : string) : string
6457: Procedure GetShellLinkInfo( const LinkFile : WideString; var SLI : TShellLinkInfo)
6458: Procedure SetShellLinkInfo( const LinkFile : WideString; const SLI : TShellLinkInfo)
6459: Function RecycleFile( FileToRecycle : string) : Boolean
6460: Function JCopyFile( FromFile, ToDir : string) : Boolean
6461: Function ShellObjectTypeEnumToConst( ShellObjectType : TShellObjectType) : UINT
6462: Function ShellObjectTypeConstToEnum( ShellObjectType : UINT) : TShellObjectType
6463: Function QueryServiceConfig2A( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
        DWORD; var pcbBytesNeeded : DWORD) : BOOL
6464: Function QueryServiceConfig2W( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
        DWORD; var pcbBytesNeeded : DWORD) : BOOL
6465: Function QueryServiceConfig2( hService : SC_HANDLE; dwInfoLevel : DWORD; lpBuffer : LPBYTE; cbBufSize :
        DWORD; var pcbBytesNeeded : DWORD) : BOOL
6466: Function EnumServicesStatusExA(hSCManager: SC_HANDLE; InfoLevel: SC_ENUM_TYPE; dwServiceType: DWORD;
        dwServiceState: DWORD; lpServices:LPBYTE;cbBufSize:DWORD;var pcbBytesNeeded,lpServicesReturned,
        lpResumeHandle:DWORD;pszGroupName: LP
6467: Function EnumServicesStatusExW( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
        dwServiceState : DWORD; lpServices:LPBYTE; cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
        lpResumeHandle:DWORD; pszGroupNam
6468: Function EnumServicesStatusEx( hSCManager : SC_HANDLE; InfoLevel : SC_ENUM_TYPE; dwServiceType : DWORD;
        dwServiceState : DWORD; lpServices :LPBYTE;cbBufSize:DWORD; var pcbBytesNeeded,lpServicesReturned,
        lpResumeHandle:DWORD; pszGroupName
6469: Function ConvertSidToStringSid( sid : PSID; var stringSid : LPWSTR) : BOOL
6470:
6471: ***** unit uPSI_JclPeImage;
6472:
6473: Function IsValidPeFile( const FileName : TFileName) : Boolean
6474: // Function PeGetNtHeaders( const FileName : TFileName; var NtHeaders : TImageNtHeaders) : Boolean
6475: Function PeCreateNameHintTable( const FileName : TFileName) : Boolean
6476: Function PeRebaseImage(const ImageName: TFileName; NewBase : DWORD; TimeStamp : DWORD; MaxNewSize :
        DWORD) : TJclRebaseImageInfo
6477: Function PeVerifyChecksum( const FileName : TFileName) : Boolean
6478: Function PeClearChecksum( const FileName : TFileName) : Boolean
6479: Function PeUpdateChecksum( const FileName : TFileName) : Boolean
6480: Function PeDoesExportFunction(const FileName:TFileName;const
        FuncName:string;Options:TJclSmartCompOptions):Bool;
6481: Function PeIsExportFunctionForwardedEx( const FileName : TFileName; const FunctionName : string; var
        ForwardedName : string; Options : TJclSmartCompOptions) : Boolean
6482: Function PeIsExportFunctionForwarded( const FileName : TFileName; const FunctionName : string; Options :
        TJclSmartCompOptions) : Boolean
6483: Function PeDoesImportFunction( const FileName : TFileName; const FunctionName : string; const LibraryName
        : string; Options : TJclSmartCompOptions) : Boolean
6484: Function PeDoesImportLibrary(const FileName:TFileName;const
        LibraryName:string;Recursive:Boolean):Boolean);
6485: Function PeImportedLibraries( const FileName : TFileName; const LibrariesList : TStrings; Recursive :
        Boolean; FullPathName : Boolean) : Boolean
6486: Function PeImportedFunctions( const FileName : TFileName; const FunctionsList : TStrings; const
        LibraryName : string; IncludeLibNames : Boolean) : Boolean
6487: Function PeExportedFunctions( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6488: Function PeExportedNames( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6489: Function PeExportedVariables( const FileName : TFileName; const FunctionsList : TStrings) : Boolean
6490: Function PeResourceKindNames(const FileN:TFileName;ResourceType:TJclPeResourceKind;const
        NamesList:TStrings):Boolean;
6491: Function PeBorFormNames( const FileName : TFileName; const NamesList : TStrings) : Boolean
6492: Function PeBorDependedPackages(const FileName:TFileName;PackagesList:TStrings;FullPathName,
        Descript:Bool):Bool;
6493: Function PeFindMissingImports( const FileName : TFileName; MissingImportsList : TStrings) : Boolean;
6494: Function PeFindMissingImports1( RequiredImportsList, MissingImportsList : TStrings) : Boolean;
6495: Function PeCreateRequiredImportList(const FileName: TFileName; RequiredImportsList: TStrings): Boolean;
6496: //Function PeMapImgNtHeaders( const BaseAddress : Pointer) : PImageNtHeaders

```

```

6497: //Function PeMapImgLibraryName( const BaseAddress : Pointer) : string
6498: //Function PeMapImgSections( const NtHeaders : PImageNtHeaders) : PImageSectionHeader
6499: //Function PeMapImgFindSection( const NtHeaders : PImageNtHeaders; const SectionName : string) :
PImageSectionHeader
6500: //Function PeMapImgExportedVariables(const Module: HMODULE; const VariablesList:TStrings):Boolean
6501: //Function PeMapImgResolvePackageThunk( Address : Pointer) : Pointer
6502: Function PeMapFindResource(const Module:HMODULE;const ResourceType:PChar;const ResourceName:string):
Pointer;
6503: SIRegister_TJclPeSectionStream(CL);
6504: SIRegister_TJclPeMapImgHookItem(CL);
6505: SIRegister_TJclPeMapImgHooks(CL);
6506: //Function PeDbgImgNtHeaders(ProcessHandle:THandle;BaseAddress:Pointer;var
NtHeaders:TImageNtHeaders):Boolean
6507: //Function PeDbgImgLibraryName(ProcessHandle:THandle; BaseAddress:Pointer; var Name:string):Boolean
6508: Type TJclBorUmSymbolKind','(skData,skFunction,skConstructor,skDestructor,skRTTI,skVTable)
6509: TJclBorUmSymbolModifier','( smQualified, smLinkProc )
6510: TJclBorUmSymbolModifiers',' set of TJclBorUmSymbolModifier
6511: TJclBorUmDescription',' record Kind : TJclBorUmSymbolKind; Modifiers : TJclBorUmSymbolModifiers; end
6512: TJclBorUmResult','( urOk, urNotMangled, urMicrosoft, urError )
6513: TJclPeUmResult','( umNotMangled, umBorland, umMicrosoft )
6514: Function PeBorUnmangleName( const Name : string; var Unmangled : string; var Description :
TJclBorUmDescription; var BasePos : Integer) : TJclBorUmResult;
6515: Function PeBorUnmangleName1(const Name:string;var Unmangled:string;var
Descript:TJclBorUmDescription):TJclBorUmResult;
6516: Function PeBorUnmangleName2( const Name : string; var Unmangled : string) : TJclBorUmResult;
6517: Function PeBorUnmangleName3( const Name : string) : string;
6518: Function PeIsNameMangled( const Name : string) : TJclPeUmResult
6519: Function PeUnmangleName( const Name : string; var Unmangled : string) : TJclPeUmResult
6520:
6521:
6522: //***** SysTools uPSI_StSystem; *****
6523: Function StCopyFile( const SrcPath, DestPath : AnsiString) : Cardinal
6524: Function CreateTempFile( const aFolder : AnsiString; const aPrefix : AnsiString) : AnsiString
6525: Function DeleteVolumeLabel( Drive : Char) : Cardinal
6526: //Procedure EnumerateDirectories(const
StartDir:AnsiString;FL:TStrings;SubDirs:Bool;IncludeItem:TIncludeItemFunc);
6527: //Procedure EnumerateFiles(const StartDir:AnsiString;FL:TStrings;SubDirs:Bool
IncludeItem:TIncludeItemFunc);
6528: Function FileHandlesLeft( MaxHandles : Cardinal) : Cardinal
6529: Function FileMatchesMask( const FileName, FileMask : AnsiString) : Boolean
6530: Function FileTimeToStDateTime( FileTime : LongInt) : TStDateTimeRec
6531: Function FindNthSlash( const Path : AnsiString; n : Integer) : Integer
6532: Function FlushOsBuffers( Handle : Integer) : Boolean
6533: Function GetCurrentUser : AnsiString
6534: Function GetDiskClass( Drive : Char) : DiskClass
6535: Function GetDiskInfo(Drive:Char;var ClustersAvail,TotalClusters,BytesPerSector,
SectorsPerCluster:Cardinal):Bool;
6536: Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Double;var TotalSpaceAvail:Double; var
DiskSize:Double):Boolean;
6537: Function GetDiskSpace(Drive:Char;var UserSpaceAvail:Comp;var TotalSpaceAvail:Comp;var
DiskSize:Comp):Boolean;
6538: { index 0 - FreeBytesAvailable, 1 - TotalNumberOfBytes, 2 - TotalNumberOfFreeBytes }
6539: Function getDiskSpace2(const path: String; index: integer): int64;
6540: Function GetFileCreateDate( const FileName : AnsiString) : TDateTime
6541: Function StGetFileLastAccess( const FileName : AnsiString) : TDateTime
6542: Function GetFileLastModify( const FileName : AnsiString) : TDateTime
6543: Function GetHomeFolder( aForceSlash : Boolean) : AnsiString
6544: Function GetLongPath( const APath : AnsiString) : AnsiString
6545: Function GetMachineName : AnsiString
6546: Function GetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6547: Function GetParentFolder( const APath : AnsiString; aForceSlash : Boolean) : AnsiString
6548: Function GetShortPath( const APath : AnsiString) : AnsiString
6549: Function GetSystemFolder( aForceSlash : Boolean) : AnsiString
6550: Function GetTempFolder( aForceSlash : boolean) : AnsiString
6551: Function StGetWindowsFolder( aForceSlash : boolean) : AnsiString
6552: Function GetWorkingFolder( aForceSlash : boolean) : AnsiString
6553: Function GlobalDateTimeToLocal( const UTC : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6554: Function StIsDirectory( const DirName : AnsiString) : Boolean
6555: Function IsDirectoryEmpty( const S : AnsiString) : Integer
6556: Function IsDriveReady( Drive : Char) : Boolean
6557: Function IsFile( const FileName : AnsiString) : Boolean
6558: Function IsFileArchive( const S : AnsiString) : Integer
6559: Function IsFileHidden( const S : AnsiString) : Integer
6560: Function IsFileReadOnly( const S : AnsiString) : Integer
6561: Function IsFileSystem( const S : AnsiString) : Integer
6562: Function LocalDateTimeToGlobal( const DT1 : TStDateTimeRec; MinOffset : Integer) : TStDateTimeRec
6563: Function ReadVolumeLabel( var VolName : AnsiString; Drive : Char) : Cardinal
6564: Function SameFile( const FilePath1, FilePath2 : AnsiString; var ErrorCode : Integer) : Boolean
6565: Function SetMediaID( Drive : Char; var MediaIDRec : MediaIDType) : Cardinal
6566: Procedure SplitPath( const APath : AnsiString; Parts : TStrings)
6567: Function StDateTimeToFileTime( const FileTime : TStDateTimeRec) : LongInt
6568: Function StDateTimeToUnixTime( const DT1 : TStDateTimeRec) : Longint
6569: Function UnixTimeToStDateTime( UnixTime : Longint) : TStDateTimeRec
6570: Function ValidDrive( Drive : Char) : Boolean
6571: Function WriteVolumeLabel( const VolName : AnsiString; Drive : Char) : Cardinal
6572:
6573: //*****unit uPSI_JclLANMan;*****
6574: Function CreateAccount( const Server, Username, Fullname, Password, Description, Homedir, Script : string;
const PasswordNeverExpires : Boolean) : Boolean

```



```

6575: Function CreateLocalAccount( const Username, Fullname, Password, Description, Homedir, Script : string;
const PasswordNeverExpires : Boolean) : Boolean
6576: Function DeleteAccount( const Servername, Username : string) : Boolean
6577: Function DeleteLocalAccount( Username : string) : Boolean
6578: Function CreateLocalGroup( const Server, Groupname, Description : string) : Boolean
6579: Function CreateGlobalGroup( const Server, Groupname, Description : string) : Boolean
6580: Function DeleteLocalGroup( const Server, Groupname : string) : Boolean
6581: Function GetLocalGroups( const Server : string; const Groups : TStrings) : Boolean
6582: Function GetGlobalGroups( const Server : string; const Groups : TStrings) : Boolean
6583: Function LocalGroupExists( const Group : string) : Boolean
6584: Function GlobalGroupExists( const Server, Group : string) : Boolean
6585: Function AddAccountToLocalGroup( const Accountname, Groupname : string) : Boolean
6586: Function LookupGroupName( const Server : string; const RID : TNetWellKnownRID) : string
6587: Procedure ParseAccountName( const QualifiedName : string; var Domain, UserName : string)
6588: Function IsLocalAccount( const AccountName : string) : Boolean
6589: Function TimeStampInterval( StartStamp, EndStamp : TDateTime) : integer
6590: Function GetRandomString( NumChar : cardinal) : string
6591:
6592: *****unit uPSI_cUtils;*****
6593: TypeS('TUnitType', '( utSrc, utHead, utRes, utPrj, utOther )
6594: Function cIsWinNT : boolean
6595: Procedure cFilesFromWildcard(Directory,Mask : string; var Files:TStringList;Subdirs,ShowDirs,
Multitasking:Boolean;
6596: Function cExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
6597: Function cRunAndGetOutput(Cmd,WorkDir:string; ErrFunc:TErrFunc; LineOutputFunc:TLineOutputFunc;
CheckAbortFunc : TCheckAbortFunc; ShowReturnValue : Boolean) : string
6598: Function cGetShortName( FileName : string) : string
6599: Procedure cShowError( Msg : String)
6600: Function cCommaStrToStr( s : string; formatstr : string) : string
6601: Function cIncludeQuoteIfSpaces( s : string) : string
6602: Function cIncludeQuoteIfNeeded( s : string) : string
6603: Procedure cLoadFilefromResource( const FileName : string; ms : TMemoryStream)
6604: Function cValidateFile(const FileName:string; const WorkPath:string; const CheckDirs:boolean):string;
6605: Function cBuildFilter( var value : string; const FLTStyle : TFILTERSET) : boolean;
6606: Function cBuildFilter1( var value : string; const _filters : array of string) : boolean;
6607: Function cCodeInstoStr( s : string) : string
6608: Function cStrtoCodeIns( s : string) : string
6609: Procedure cStrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6610: Function cAttrtoStr( const Attr : TSynHighlighterAttributes) : string
6611: Procedure cStrtoPoint( var pt : TPoint; value : string)
6612: Function cPointtoStr( const pt : TPoint) : string
6613: Function cListtoStr( const List : TStrings) : string
6614: Function ListtoStr( const List : TStrings) : string
6615: Procedure StrtoList( s : string; const List : TStrings; const delimiter : char)
6616: Procedure cStrtoList( s : string; const List : TStrings; const delimiter : char)
6617: Function cGetFileType( const FileName : string) : TUnitType
6618: Function cGetExTyp( const FileName : string) : TExUnitType
6619: Procedure cSetPath( Add : string; const UseOriginal : boolean)
6620: Function cExpandFileto( const FileName : string; const BasePath : string) : string
6621: Function cFileSamePath( const FileName : string; const TestPath : string) : boolean
6622: Procedure cCloneMenu( const FromMenu : TMenuItem; ToMenu : TMenuItem)
6623: Function cGetLastPos( const SubStr : string; const S : string) : integer
6624: Function cGenMakePath( FileName : String) : String;
6625: Function cGenMakePath2( FileName : String) : String
6626: Function cGenMakePath1( FileName : String; EscapeSpaces, EncloseInQuotes : Boolean) : String;
6627: Function cGetRealPath( BrokenFileName : String; Directory : String) : String
6628: Function cCalcMod( Count : Integer) : Integer
6629: Function cGetVersionString( FileName : string) : string
6630: Function cCheckChangeDir( var Dir : string) : boolean
6631: Function cGetAssociatedProgram(const Extension:string; var Filename,Description: string):boolean
6632: Function cIsNumeric( s : string) : boolean
6633: Procedure StrtoAttr( var Attr : TSynHighlighterAttributes; Value : string)
6634: Function AttrtoStr( const Attr : TSynHighlighterAttributes) : string
6635: Function GetFileType( const FileName : string) : TUnitType
6636: Function Atoi(const aStr: string): integer
6637: Function Itoa(const aint: integer): string
6638:
6639:
6640: procedure SIRegister_cHTTPUtils(CL: TPSPascalCompiler);
6641: begin
6642: FindClass('TOBJECT'),'EHTTP
6643: FindClass('TOBJECT'),'EHTTTParser
6644: //AnsiCharSet', 'set of AnsiChar
6645: AnsiStringArray', 'array of AnsiString
6646: THTTPProtocolEnum', '( hpNone, hpCustom, hpHTTP, hpHTTPS )
6647: THTTPVersionEnum', '( hvNone, hvCustom, hvHTTPL0, hvHTTPL1 )
6648: THTTPVersion', 'record Version : THTTPVersionEnum; Protocol : TH
6649: +TTPProtocolEnum; CustomProtocol : AnsiString; CustomMajVersion : Integer; '
6650: +CustomMinVersion : Integer; end
6651: THTTPHeaderNameEnum', '( hntCustom, hntHost, hntContentType, hnt'
6652: +ContentLength, hntContentTransferEncoding, hntContentLocation, hntContentL'
6653: +language, hntContentEncoding, hntTransferEncoding, hntDate, hntServer, hntU'
6654: +serAgent, hntLocation, hntConnection, hntExpires, hntCacheControl, hntSetC'
6655: +ookie, hntCookie, hntAuthorization, hntVia, hntWarning, hntContentRange, h'
6656: +ntXForwardedFor, hntPragma, hntXPoweredBy, hntWWWAuthenticate, hntLastModi'
6657: +fied, hntETag, hntProxyAuthorization, hntReferer, hntAge, hntAcceptRanges,'
6658: +hntAcceptEncoding, hntAcceptLanguage, hntAcceptCharset, hntIfModifiedSinc'
6659: +e, hntIfUnmodifiedSince, hntRetryAfter, hntUpgrade, hntStatus, hntProxyCon'
6660: +nection, hntOrigin, hntKeepAlive )

```

```

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

```

```

6750: //PHTTPResponseHeader', '^THTTPResponseHeader // will not work
6751: THTTPResponse', 'record StartLine : THTTPResponseStartLine; Head'
6752: + 'er : THTTPResponseHeader; HeaderComplete : Boolean; HasContent : Boolean; end
6753: Function HTTPMessageHasContent( const H : THTTPCommonHeaders ) : Boolean
6754: Procedure InitHTTPRequest( var A : THTTPRequest )
6755: Procedure InitHTTPResponse( var A : THTTPResponse )
6756: Procedure ClearHTTPVersion( var A : THTTPVersion )
6757: Procedure ClearHTTPContentLength( var A : THTTPContentLength )
6758: Procedure ClearHTTPContentType( var A : THTTPContentType )
6759: Procedure ClearHTTPDateField( var A : THTTPDateField )
6760: Procedure ClearHTTPTransferEncoding( var A : THTTPTransferEncoding )
6761: Procedure ClearHTTPConnectionField( var A : THTTPConnectionField )
6762: Procedure ClearHTTPPageField( var A : THTTPPageField )
6763: Procedure ClearHTTPContentEncoding( var A : THTTPContentEncoding )
6764: Procedure ClearHTTPContentEncodingField( var A : THTTPContentEncodingField )
6765: Procedure ClearHTTPContentRangeField( var A : THTTPContentRangeField )
6766: Procedure ClearHTTPSetCookieField( var A : THTTPSetCookieField )
6767: Procedure ClearHTTPCommonHeaders( var A : THTTPCommonHeaders )
6768: //Procedure ClearHTTPFixedHeaders( var A : THTTPFixedHeaders )
6769: Procedure ClearHTTPCustomHeaders( var A : THTTPCustomHeaders )
6770: Procedure ClearHTTPCookieField( var A : THTTPCookieField )
6771: Procedure ClearHTTPMethod( var A : THTTPMethod )
6772: Procedure ClearHTTPRequestStartLine( var A : THTTPRequestStartLine )
6773: Procedure ClearHTTPRequestHeader( var A : THTTPRequestHeader )
6774: Procedure ClearHTTPRequest( var A : THTTPRequest )
6775: Procedure ClearHTTPResponseStartLine( var A : THTTPResponseStartLine )
6776: Procedure ClearHTTPResponseHeader( var A : THTTPResponseHeader )
6777: Procedure ClearHTTPResponse( var A : THTTPResponse )
6778: THTTPStringOption', '( hsoNone )
6779: THTTPStringOptions', 'set of THTTPStringOption
6780: FindClass('TOBJECT'),'TAnsiStringBuilder
6781:
6782: Procedure BuildStrHTTPVersion(const A:THTTPVersion; const B:TAnsiStringBuilder;const
P:THTTPStringOptions);
6783: Procedure BuildStrHTTPContentLengthValue( const A : THTTPContentLength; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6784: Procedure BuildStrHTTPContentLength( const A : THTTPContentLength; const B : TAnsiStringBuilder; const P
: THTTPStringOptions)
6785: Procedure BuildStrHTTPContentTypeValue( const A : THTTPContentType; const B : TAnsiStringBuilder; const P
: THTTPStringOptions)
6786: Procedure BuildStrHTTPContentType(const A:THTTPContentType;const B:TAnsiStringBuilder; const
P:THTTPStringOptions)
6787: Procedure BuildStrRFCDateTime( const DOW, Da, Mo, Ye, Ho, Mi, Se : Integer; const TZ : AnsiString; const
B : TAnsiStringBuilder; const P : THTTPStringOptions)
6788: Procedure BuildStrHTTPDateFieldValue( const A : THTTPDateField; const B : TAnsiStringBuilder; const P :
THTTPStringOptions)
6789: Procedure BuildStrHTTPDateField(const A:THTTPDateField;const B:TAnsiStringBuilder;const
P:THTTPStringOptions);
6790: Procedure BuildStrHTTPTransferEncodingValue( const A : THTTPTransferEncoding; const B :
TAnsiStringBuilder; const P : THTTPStringOptions)
6791: Procedure BuildStrHTTPTransferEncoding( const A : THTTPTransferEncoding; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6792: Procedure BuildStrHTTPContentRangeField( const A : THTTPContentRangeField; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6793: Procedure BuildStrHTTPConnectionFieldValue( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6794: Procedure BuildStrHTTPConnectionField( const A : THTTPConnectionField; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6795: Procedure BuildStrHTTPPageField(const A:THTTPPageField;const B:TAnsiStringBuilder;const
P:THTTPStringOptions);
6796: Procedure BuildStrHTTPContentEncoding( const A : THTTPContentEncoding; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6797: Procedure BuildStrHTTPContentEncodingField(const A:THTTPContentEncodingField;const
B:TAnsiStringBuilder;const P:THTTPStringOptions)
6798: Procedure BuildStrHTTPProxyConnectionField(const A : THTTPConnectionField; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6799: Procedure BuildStrHTTPCommonHeaders( const A : THTTPCommonHeaders; const B : TAnsiStringBuilder; const P
: THTTPStringOptions)
6800: Procedure BuildStrHTTPFixedHeaders(const A:THTTPFixedHeaders;const B:TAnsiStrBuilder;const
P:THTTPStringOptions)
6801: Procedure BuildStrHTTPCustomHeaders( const A : THTTPCustomHeaders; const B : TAnsiStringBuilder; const P
: THTTPStringOptions)
6802: Procedure BuildStrHTTPSetCookieFieldValue( const A : THTTPSetCookieField; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6803: Procedure BuildStrHTTPCookieFieldValue( const A : THTTPCookieField; const B : TAnsiStringBuilder; const P
: THTTPStringOptions)
6804: Procedure BuildStrHTTPCookieField(const A:THTTPCookieField;const B:TAnsiStringBuilder;const
P:THTTPStringOptions);
6805: Procedure BuildStrHTTPMethod( const A : THTTPMethod; const B : TAnsiStringBuilder; const P :
THTTPStringOptions)
6806: Procedure BuildStrHTTPRequestStartLine( const A : THTTPRequestStartLine; const B : TAnsiStringBuilder;
const P : THTTPStringOptions)
6807: Procedure BuildStrHTTPRequestHeader(const A:THTTPRequestHeader;const B:TAnsiStringBuilder;const
P:THTTPStringOptions);
6808: Procedure BuildStrHTTPRequest( const A : THTTPRequest; const B : TAnsiStringBuilder; const P :
THTTPStringOptions)
6809: Procedure BuildStrHTTPResponseCookieFieldArray( const A : THTTPSetCookieFieldArray; const B :
TAnsiStringBuilder; const P : THTTPStringOptions)
6810: Procedure BuildStrHTTPResponseStartLine(const A:THTTPResponseStartLine;const B:TAnsiStrBldr;const P
THTTPStrOptions);

```

```

6811: Procedure BuildStrHTTPResponseHeader(const A:THTTPrespHeader;const B:TAnsiStrBuilder;const
P:THTTPStringOptions);
6812: Procedure BuildStrHTTPResponse(const A:THTTPResponse; const B:TAnsiStringBuilder; const
P:THTTPStringOptions);
6813: Function HTTPContentTypeValueToStr( const A : THTTPContentType) : AnsiString
6814: Function HTTPSetCookieFieldValueToStr( const A : THTTPSetCookieField) : AnsiString
6815: Function HTTPCookieFieldValueToStr( const A : THTTPCookieField) : AnsiString
6816: Function HTTPMethodToStr( const A : THTTPMethod) : AnsiString
6817: Function HTTPRequestToStr( const A : THTTPRequest) : AnsiString
6818: Function HTTPResponseToStr( const A : THTTPResponse) : AnsiString
6819: Procedure PrepareCookie( var A:THTTPCookieField;const B:THTTPSetCookieFieldArray; const
Domain:AnsiString; const Secure : Boolean);
6820: THTTPParserHeaderParseFunc', 'Function ( const HeaderName : THTT
+ 'PHeaderNameEnum; const HeaderPtr : ___Pointer) : Boolean
6821: SIRegister_THTTPParser(CL);
6822: FindClass('TOBJECT'),'THTTPContentDecoder
6823: THTTPContentDecoderProc', 'Procedure ( const Sender : THTTPContentDecoder)
6824: THTTPContentDecoderContentType', '( crctFixedSize, crctChunked, crctUnsize )
6825: THTTPContentDecoderChunkState', '( crcsChunkHeader, crcsContent,
+ ' crcsContentCRLF, crcsTrailer, crcsFinished )
6826: THTTPContentDecoderLogEvent', 'Procedure ( const Sender : THTTPContentDecoder; const LogMsg : String)
6827: SIRegister_THTTPContentDecoder(CL);
6828: THTTPContentReaderMechanism', '( hcrnEvent, hcrnString, hcrnStream, hcrnFile )
6829: FindClass('TOBJECT'),'THTTPContentReader
6830: THTTPContentReaderProc', 'Procedure ( const Sender : THTTPContentReader)
6831: THTTPContentReaderLogEvent', 'Procedure(const Sender:THTTPContentReader;const LogMsg:String;const
LogLevel:Int;
6832: SIRegister_THTTPContentReader(CL);
6833: THTTPContentWriterMechanism', '( hctmEvent, hctmString, hctmStream, hctmFile )
6834: FindClass('TOBJECT'),'THTTPContentWriter
6835: THTTPContentWriterLogEvent', 'Procedure (const Sender : THTTPContentWriter;const LogMsg:AnsiString);
6836: SIRegister_THTTPContentWriter(CL);
6837: Procedure SelfTestcHTTUtils
6838: end;
6839:
6840:
6841:
6842: (*-----*)
6843: procedure SIRegister_cTLSUtils(CL: TPSPascalCompiler);
6844: begin
6845: 'TLSLibraryVersion','String').SetString( '1.00
6846: 'TLSError_None','LongInt').SetInt( 0);
6847: 'TLSError_InvalidBuffer','LongInt').SetInt( 1);
6848: 'TLSError_InvalidParameter','LongInt').SetInt( 2);
6849: 'TLSError_InvalidCertificate','LongInt').SetInt( 3);
6850: 'TLSError_InvalidState','LongInt').SetInt( 4);
6851: 'TLSError_DecodeError','LongInt').SetInt( 5);
6852: 'TLSError_BadProtocol','LongInt').SetInt( 6);
6853: Function TLSErrorMessage( const TLSError : Integer) : String
6854: SIRegister_ETLSError(CL);
6855: TTLSProtocolVersion', 'record major : Byte; minor : Byte; end
6856: PTLSPProtocolVersion', '^TTLSProtocolVersion // will not work
6857: Procedure InitSSLProtocolVersion30( var A : TTLSProtocolVersion)
6858: Procedure InitTLSProtocolVersion10( var A : TTLSProtocolVersion)
6859: Procedure InitTLSProtocolVersion11( var A : TTLSProtocolVersion)
6860: Procedure InitTLSProtocolVersion12( var A : TTLSProtocolVersion)
6861: Function IsTLSProtocolVersion( const A, B : TTLSProtocolVersion) : Boolean
6862: Function IsSSL2( const A : TTLSProtocolVersion) : Boolean
6863: Function IsSSL3( const A : TTLSProtocolVersion) : Boolean
6864: Function IsTLS10( const A : TTLSProtocolVersion) : Boolean
6865: Function IsTLS11( const A : TTLSProtocolVersion) : Boolean
6866: Function IsTLS12( const A : TTLSProtocolVersion) : Boolean
6867: Function IsTLS10OrLater( const A : TTLSProtocolVersion) : Boolean
6868: Function IsTLS11OrLater( const A : TTLSProtocolVersion) : Boolean
6869: Function IsTLS12OrLater( const A : TTLSProtocolVersion) : Boolean
6870: Function IsFutureTLSVersion( const A : TTLSProtocolVersion) : Boolean
6871: Function IsKnownTLSVersion( const A : TTLSProtocolVersion) : Boolean
6872: Function TLSProtocolVersionToStr( const A : TTLSProtocolVersion) : String
6873: Function TLSProtocolVersionName( const A : TTLSProtocolVersion) : String
6874: PTLSPRandom', '^TTLSRandom // will not work
6875: Procedure InitTLSPRandom( var Random : TTLSRandom)
6876: Function TLSPRandomToStr( const Random : TTLSRandom) : AnsiString
6877: 'TLSSessionIDMaxLen','LongInt').SetInt( 32);
6878: Procedure InitTLSSessionID( var SessionID : TTLSSessionID; const A : AnsiString)
6879: Function EncodeTLSSessionID( var Buffer:string;const Size:Integer;const SessionID:TTLSSessionID):Integer;
6880: Function DecodeTLSSessionID(const Buffer:string;const Size:Integer;var SessionID:TTLSSessionID):Integer;
6881: TTLSSignatureAndHashAlgorithm', 'record Hash : TTLSHashAlgorithm
+ 'Signature : TTLSSignatureAlgorithm; end
6882: // PTLSSignatureAndHashAlgorithm', '^TTLSSignatureAndHashAlgorithm +// will not work
6883: TTLSSignatureAndHashAlgorithmArray', 'array of TTLSSignatureAndHashAlgorithm
6884: TTLSKeyExchangeAlgorithm', '( tlskeaNone, tlskeaNULL, tlskeaDHE
+ 'DSS, tlskeaDHE_RSA, tlskeaDH_Anon, tlskeaRSA, tlskeaDH_DSS, tlskeaDH_RSA )
6885: TTLSMACAlgorithm', '( tlsmaNone, tlsmaNULL, tlsmaHMAC_MD5, tlsma
+ 'HMAC_SHA1, tlsmaHMAC_SHA256, tlsmaHMAC_SHA384, tlsmaHMAC_SHA512 )
6886: TTLSMacAlgorithmInfo', 'record Name : AnsiString; DigestSize : Integer; Supported : Boolean; end
6887: PTLSPMacAlgorithmInfo', '^TTLSMacAlgorithmInfo // will not work
6888: 'TLS_MAC_MAXDIGESTSIZE','LongInt').SetInt( 64);
6889: TTLSPRFAlgorithm', '( tlspsaSHA256 )
6890: Function tlp_MD5( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6891: Function tlp_SHA1( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString

```



```

6896: Function tlsP_SHA256( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6897: Function tlsP_SHA512( const Secret, Seed : AnsiString; const Size : Integer) : AnsiString
6898: Function tls10PRF( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6899: Function tls12PRF_SHA256( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6900: Function tls12PRF_SHA512( const Secret, ALabel, Seed : AnsiString; const Size : Integer) : AnsiString
6901: Function TLS1PRF(const ProtoVersion:TTLSProtocolVersion;const Secret,ALabel,Seed:AStrng;const
Size:Int):AStrng;
6902: Function tls10KeyBlock(const MasterSecret, ServerRandom,ClientRandom:AnsiString; const
Size:Integer):AnsiString
6903: Function tls12SHA256KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
Size:Int):AnsiString;
6904: Function tls12SHA512KeyBlock(const MasterSecret,ServerRandom,ClientRandom: AnsiString;const
Size:Int):AnsiString;
6905: Function TLSKeyBlock( const ProtocolVersion : TTLSProtocolVersion; const MasterSecret, ServerRandom,
ClientRandom : AnsiString; const Size : Integer) : AnsiString
6906: Function tls10MasterSecret(const PreMasterSecret,ClientRandom, ServerRandom:AnsiString) : AnsiString;
6907: Function tls12SHA256MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString):AnsiString;
6908: Function tls12SHA512MasterSecret(const PreMasterSecret,ClientRandom,ServerRandom:AnsiString): AnsiString;
6909: Function TLSPMasterSecret( const ProtocolVersion: TTLSProtocolVersion;const PreMasterSecret,ClientRandom,
ServerRandom:AnsiString) : AnsiString
6910: TLSKeys', 'record KeyBlock : AnsiString; ClientMACKey : AnsiStr'
6911: +ing; ServerMACKey : AnsiString; ClientEncKey : AnsiString; ServerEncKey : '
6912: +AnsiString; ClientIV : AnsiString; ServerIV : AnsiString; end
6913: Procedure GenerateTLSKeys( const ProtocolVersion : TTLSProtocolVersion; const MACKeyBits, CipherKeyBits,
IVBits : Integer; const MasterSecret, ServerRandom, ClientRandom : AnsiString; var TLSKeys : TLSKeys)
6914: Procedure GenerateFinalTLSKeys( const ProtocolVersion : TTLSProtocolVersion; const IsExportable :
Boolean; const ExpandedKeyBits : Integer; const ServerRandom, ClientRandom : AnsiString; var TLSKeys :
TLSKeys)
6915: 'TLS_PLAINTEXT_FRAGMENT_MAXSIZE','LongInt').SetInt( 16384 - 1);
6916: 'TLS_COMPRESSED_FRAGMENT_MAXSIZE','LongInt').SetInt( 16384 + 1024);
6917: Procedure SelfTestcTLSUtils
6918: end;
6919:
6920: (*-----*)
6921: procedure SIRegister_Reversi(CL: TPSPascalCompiler);
6922: begin
6923:   sPosData', 'record corner : boolean; square2x2 : boolean; edge : '
6924:   + ' boolean; stable : integer; internal : integer; disks : integer; mx : integer; my : integer; end
6925:   // pBoard', 'tBoard // will not work
6926:   Function rCalculateData( cc : byte; cx, cy : integer) : sPosData
6927:   Function rCheckMove( color : byte; cx, cy : integer) : integer
6928:   //Function rDoStep( data : pBoard) : word
6929:   Function winExecAndWait( const sAppPath : string; wVisible : word) : boolean
6930: end;
6931:
6932: procedure SIRegister_IWDBCommon(CL: TPSPascalCompiler);
6933: begin
6934:   Function InEditMode( ADataset : TDataset) : Boolean
6935:   Function CheckDataSource( ADataSource : TDataSource) : Boolean;
6936:   Function CheckDataSource1(ADataSource:TDataSource;const AFieldName:string;var VField:TField):boolean;
6937:   Function GetFieldText( AField : TField) : String
6938: end;
6939:
6940: procedure SIRegister_SortGrid(CL: TPSPascalCompiler);
6941: begin
6942:   TPrintMode', '( pmPrint, pmPreview, pmPageCount )
6943:   TMyPrintRange', '( prAll, prSelected )
6944:   TSortStyle', '( ssAutomatic, ssNormal, ssNumeric, ssNumericExten'
6945:   + 'ded, ssDateTime, ssTime, ssCustom )
6946:   TSortDirection', '( sdAscending, sdDescending )
6947:   TSetChecked', 'Procedure ( Sender : TObject; ACol, ARow : integer; State : Boolean)
6948:   TGetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integ'
6949:   + 'er; var Strs : TStringList; var Width, Height : integer; var Sorted : Boolean)
6950:   TSetCombobox', 'Procedure ( Sender : TObject; ACol, ARow : integer; Str : String)
6951:   TSetEllipsis', 'Procedure ( Sender : TObject; ACol, ARow : integer)
6952:   SIRegister_TSortOptions(CL);
6953:   SIRegister_TPrintOptions(CL);
6954:   TSortedListEntry', 'record Str : String; RowNum : integer; SortOption : TSortOptions; end
6955:   SIRegister_TSortedList(CL);
6956:   TCellBevelStyle', '( cbNone, cbRaised, cbLowered )
6957:   TCellBevel', 'record Style: TCellBevelStyle; UpperLeftColor: TColor; LowerRightColor : TColor; end
6958:   TVertAlignment', '( taTopJustify, taBottomJustify, taMiddle )
6959:   TFormatOptions', 'record Brush : TBrush; Font : TFont; Alignment'
6960:   + 'Horz : TAlignment; AlignmentVert : TVertAlignment; Bevel : TCellBevel; HideText : Boolean; end
6961:   SIRegister_TFontSetting(CL);
6962:   SIRegister_TFontList(CL);
6963:   AddTypeS(TFormatDrawCellEvent', 'Procedure ( Sender : TObject; Col, Row : '
6964:   + ' integer;State:TGridDrawState;var FormatOptions:TFormatOptions;var CheckBox,Combobox,Ellipsis:Bool);
6965:   TSetFilterEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
6966:   TSearchEvent', 'Procedure ( ARows : TStrings; var Accept : Boolean)
6967:   TUpdateGridEvent', 'Procedure ( Sender : TObject; ARow : integer)
6968:   TSizeChangeEvent', 'Procedure ( Sender : TObject; OldColCount, OldRowCount : integer)
6969:   TBeginSortEvent', 'Procedure ( Sender : TObject; var Col : integer)
6970:   TEndSortEvent', 'Procedure ( Sender : TObject; Col : integer)
6971:   TGetSortStyleEvent', 'Procedure ( Sender : TObject; Col : integer)
6972:   + 'r; var SortStyle : TSortStyle)
6973:   TCellValidateEvent', 'Procedure ( Sender : TObject; ACol, ARow : '
6974:   + ' integer; const OldValue : string; var NewValue : String; var Valid : Boolean)
6975:   SIRegister_TSortGrid(CL);

```

```

6976: Function ExtendedCompare( const Str1, Str2 : String) : Integer
6977: Function NormalCompare( const Str1, Str2 : String) : Integer
6978: Function DateTimeCompare( const Str1, Str2 : String) : Integer
6979: Function NumericCompare( const Str1, Str2 : String) : Integer
6980: Function TimeCompare( const Str1, Str2 : String) : Integer
6981: //Function Compare( Item1, Item2 : Pointer) : Integer
6982: end;
6983:
6984: ***** procedure Register_IB(CL: TPSPascalCompiler);
6985: Procedure IBAlloc( var P, OldSize, NewSize : Integer)
6986: Procedure IBError( ErrMess : TIBClientError; const Args : array of const)
6987: Procedure IBDataBaseError
6988: Function StatusVector : PISC_STATUS
6989: Function StatusVectorArray : PStatusVector
6990: Function CheckStatusVector( ErrorCodes : array of ISC_STATUS) : Boolean
6991: Function StatusVectorAsText : string
6992: Procedure SetIBDataBaseErrorMessages( Value : TIBDataBaseErrorMessages)
6993: Function GetIBDataBaseErrorMessages : TIBDataBaseErrorMessages
6994:
6995:
6996: //*****unit uPSI_BoldUtils;*****
6997: Function CharCount( c : char; const s : string) : integer
6998: Function BoldNamesEqual( const name1, name2 : string) : Boolean
6999: Procedure BoldAppendToStrings(strings: TStrings; const aString: string; const ForceNewLine: Boolean)
7000: Function BoldSeparateStringList(strings: TStringList; const Separator, PreString, PostString: String): String
7001: Function BoldSeparatedAppend( const S1, S2 : string; const Separator : string) : string
7002: Function BoldTrim( const S : string) : string
7003: Function BoldIsPrefix( const S, Prefix : string) : Boolean
7004: Function BoldStrEqual( P1, P2 : PChar; Len : integer) : Boolean
7005: Function BoldStrAnsiEqual( P1, P2 : PChar; Len : integer) : Boolean
7006: Function BoldAnsiEqual( const S1, S2 : string) : Boolean
7007: Function BoldStrStringEqual( const S1 : string; P2 : PChar; Len : integer) : Boolean
7008: Function BoldCaseIndependentPos( const Substr, S : string) : Integer
7009: //Procedure EnumToStrings( aTypeInfo : pTypeInfo; Strings : TStrings)
7010: Function CapitalisedToSpaced( Capitalised : String) : String
7011: Function SpacedToCapitalised( Spaced : String) : String
7012: Function BooleanToString( BoolValue : Boolean) : String
7013: Function StringToBoolean( StrValue : String) : Boolean
7014: Function GetUpperLimitForMultiplicity( const Multiplicity : String) : Integer
7015: Function GetLowerLimitForMultiplicity( const Multiplicity : String) : Integer
7016: Function StringListToVarArray( List : TStringList) : variant
7017: Function IsLocalMachine( const Machinename : WideString) : Boolean
7018: Function GetComputerNameStr : string
7019: Function TimeStampComp( const Time1, Time2 : TTimeStamp) : Integer
7020: Function BoldStrToDateFmt( const S: string; const DateFormat: string; const DateSeparatorChar: char): TDateTime
7021: Function BoldDateToStrFmt( const aDate: TDateTime; DateFormat: string; const DateSeparatorChar: char): String;
7022: Function BoldParseFormattedDateList( const value: String; const formats: TStrings; var Date: TDateTime): Boolean;
7023: Function BoldParseFormattedDate( const value: String; const formats: array of string; var
Date: TDateTime): Boolean;
7024: Procedure EnsureTrailing( var Str : String; ch : char)
7025: Function BoldDirectoryExists( const Name : string) : Boolean
7026: Function BoldForceDirectories( Dir : string) : Boolean
7027: Function BoldRootRegistryKey : string
7028: Function GetModuleFileNameAsString( IncludePath : Boolean) : string
7029: Function BoldVariantToStrings( V : OleVariant; Strings : TStrings) : Integer
7030: Function LogicalAnd( A, B : Integer) : Boolean
7031: record TByHandleFileInformation dwFileAttributes : DWORD; '
7032: +ftCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime '
7033: +': TFileTime; dwVolumeSerialNumber : DWORD; nFileSizeHigh : DWORD; nFileSiz'
7034: +eLow : DWORD; nNumberOfLinks : DWORD; nFileIndexHigh : DWORD; nFileIndexLow : DWORD; end
7035: Function GetFileInformationByHandle(hFile: THandle; var lpFileInformation: TByHandleFileInformation): BOOL;
7036: Function IsFirstInstance : Boolean
7037: Procedure ActivateFirst( AString : PChar)
7038: Procedure ActivateFirstCommandLine
7039: function MakeAckPkt( const BlockNumber: Word): string;
7040: procedure SendError(UDPPBase: TIdUDPPBase; APeerIP: string; const APort: Int; const ErrNumber: Word; ErrorString:
string);
7041: procedure SendError(UDPClient: TIdUDPClient; const ErrNumber: Word; ErrorString: string); overload;
7042: procedure SendError(UDPPBase: TIdUDPPBase; APeerIP: string; const APort: Integer; E: Exception); overload;
7043: procedure SendError(UDPClient: TIdUDPClient; E: Exception); overload;
7044: function IdStrToWord( const Value: String): Word;
7045: function IdWordToStr( const Value: Word): WordStr;
7046: Function HasInstructionSet( const InstructionSet : TCPUIInstructionSet) : Boolean
7047: Function CPUFeatures : TCPUFeatures
7048:
7049:
7050: procedure SIRegister_xrtl_util_CPUUtils(CL: TPSPascalCompiler);
7051: begin
7052: AddTypes('TXRTLBitIndex', 'Integer
7053: Function XRTLSwapBits( Data : Cardinal; Bit1Index, Bit2Index : TXRTLBitIndex) : Cardinal
7054: Function XRTLBitTest( Data : Cardinal; BitIndex : TXRTLBitIndex) : Boolean
7055: Function XRTLBitSet( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7056: Function XRTLBitReset( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7057: Function XRTLBitComplement( Data : Cardinal; BitIndex : TXRTLBitIndex) : Cardinal
7058: Function XRTLSwapHiLo16( X : Word) : Word
7059: Function XRTLSwapHiLo32( X : Cardinal) : Cardinal
7060: Function XRTLSwapHiLo64( X : Int64) : Int64
7061: Function XRTLROL32( A, S : Cardinal) : Cardinal
7062: Function XRTLROL32( A, S : Cardinal) : Cardinal

```

```

7063: Function XRTLROL16( A : Word; S : Cardinal) : Word
7064: Function XRTLROL16( A : Word; S : Cardinal) : Word
7065: Function XRTLROL8( A : Byte; S : Cardinal) : Byte
7066: Function XRTLROL8( A : Byte; S : Cardinal) : Byte
7067: //Procedure XRTLXorBlock( I1, I2, O1 : PByteArray; Len : integer)
7068: //Procedure XRTLIncBlock( P : PByteArray; Len : integer)
7069: Procedure XRTLUMul64( const A, B : Integer; var Mull, MulH : Integer)
7070: Function XRTLPopulation( A : Cardinal) : Cardinal
7071: end;
7072:
7073: Function XRTLURLDecode( const ASrc : WideString) : WideString
7074: Function XRTLURLEncode( const ASrc : WideString) : WideString
7075: Function XRTLURINormalize( const AURI : WideString) : WideString
7076: Procedure XRTLURIParse(const AURI:Widestring;var VProtocol,VHost,VPath,VDocument,VPort,VBookmark,VUserName,
VPassword : WideString)
7077: Function XRTLExtractLongPathName(APath: string): string;
7078:
7079: procedure SIRegister_cFundamentUtils(CL: TPSPascalCompiler);
7080: begin
7081:   AddTypeS('Int8', 'ShortInt
7082:   AddTypeS('Int16', 'SmallInt
7083:   AddTypeS('Int32', 'LongInt
7084:   AddTypeS('UInt8', 'Byte
7085:   AddTypeS('UInt16', 'Word
7086:   AddTypeS('UInt32', 'LongWord
7087:   AddTypeS('UInt64', 'Int64
7088:   AddTypeS('Word8', 'UInt8
7089:   AddTypeS('Word16', 'UInt16
7090:   AddTypeS('Word32', 'UInt32
7091:   AddTypeS('Word64', 'UInt64
7092:   AddTypeS('LargeInt', 'Int64
7093:   AddTypeS('NativeInt', 'Integer
7094:   AddTypeS('NativeUInt', 'Cardinal
7095:   Const('BitsPerByte','LongInt').SetInt( 8);
7096:   Const('BitsPerWord','LongInt').SetInt( 16);
7097:   Const('BitsPerLongWord','LongInt').SetInt( 32);
7098:   //Const('BitsPerCardinal','LongInt').SetInt( BytesPerCardinal * 8);
7099:   //Const('BitsPerNativeWord','LongInt').SetInt( BytesPerNativeWord * 8);
7100:   Function MinI( const A, B : Integer) : Integer
7101:   Function MaxI( const A, B : Integer) : Integer
7102:   Function MinC( const A, B : Cardinal) : Cardinal
7103:   Function MaxC( const A, B : Cardinal) : Cardinal
7104:   Function SumClipI( const A, I : Integer) : Integer
7105:   Function SumClipC( const A : Cardinal; const I : Integer) : Cardinal
7106:   Function InByteRange( const A : Int64) : Boolean
7107:   Function InWordRange( const A : Int64) : Boolean
7108:   Function InLongWordRange( const A : Int64) : Boolean
7109:   Function InShortIntRange( const A : Int64) : Boolean
7110:   Function InSmallIntRange( const A : Int64) : Boolean
7111:   Function InLongIntRange( const A : Int64) : Boolean
7112:   AddTypeS('Bool8', 'ByteBool
7113:   AddTypeS('Bool16', 'WordBool
7114:   AddTypeS('Bool32', 'LongBool
7115:   AddTypeS('TCompareResult', '( crLess, crEqual, crGreater, crUndefined )
7116:   AddTypeS('TCompareResultSet', 'set of TCompareResult
7117:   Function ReverseCompareResult( const C : TCompareResult) : TCompareResult
7118:   Const('MinSingle','Single').setExtended( 1.5E-45);
7119:   Const('MaxSingle','Single').setExtended( 3.4E+38);
7120:   Const('MinDouble','Double').setExtended( 5.0E-324);
7121:   Const('MaxDouble','Double').setExtended( 1.7E+308);
7122:   Const('MinExtended','Extended').setExtended(3.4E-4932);
7123:   Const('MaxExtended','Extended').setExtended(1.1E+4932);
7124:   Const('MinCurrency','Currency').SetExtended( - 922337203685477.5807);
7125:   Const('MaxCurrency','Currency').SetExtended( 922337203685477.5807);
7126:   Function MinF( const A, B : Float) : Float
7127:   Function MaxF( const A, B : Float) : Float
7128:   Function ClipF( const Value : Float; const Low, High : Float) : Float
7129:   Function InSingleRange( const A : Float) : Boolean
7130:   Function InDoubleRange( const A : Float) : Boolean
7131:   Function InCurrencyRange( const A : Float) : Boolean;
7132:   Function InCurrencyRange1( const A : Int64) : Boolean;
7133:   Function FloatExponentBase2( const A : Extended; var Exponent : Integer) : Boolean
7134:   Function FloatExponentBase10( const A : Extended; var Exponent : Integer) : Boolean
7135:   Function FloatIsInfinity( const A : Extended) : Boolean
7136:   Function FloatIsNaN( const A : Extended) : Boolean
7137:   Const('SingleCompareDelta','Extended').setExtended( 1.0E-34);
7138:   Const('DoubleCompareDelta','Extended').setExtended( 1.0E-280);
7139:   Const('ExtendedCompareDelta','Extended').setExtended( 1.0E-4400);
7140:   Const('DefaultCompareDelta','Extended').SetExtended( 1.0E-34);
7141:   Function FloatZero( const A : Float; const CompareDelta : Float) : Boolean
7142:   Function FloatOne( const A : Float; const CompareDelta : Float) : Boolean
7143:   Function FloatsEqual( const A, B : Float; const CompareDelta : Float) : Boolean
7144:   Function FloatsCompare( const A, B : Float; const CompareDelta : Float) : TCompareResult
7145:   Const('SingleCompareEpsilon','Extended').setExtended( 1.0E-5);
7146:   Const('DoubleCompareEpsilon','Extended').setExtended( 1.0E-13);
7147:   Const('ExtendedCompareEpsilon','Extended').setExtended( 1.0E-17);
7148:   Const('DefaultCompareEpsilon','Extended').setExtended( 1.0E-10);
7149:   Function ApproxEqual( const A, B : Extended; const CompareEpsilon : Double) : Boolean
7150:   Function ApproxCompare( const A, B : Extended; const CompareEpsilon : Double): TCompareResult

```

```

7151: Function cClearBit( const Value : LongWord) : LongWord
7152: Function cSetBit( const Value, BitIndex : LongWord) : LongWord
7153: Function cIsBitSet( const Value, BitIndex : LongWord) : Boolean
7154: Function cToggleBit( const Value, BitIndex : LongWord) : LongWord
7155: Function cIsHighBitSet( const Value : LongWord) : Boolean
7156: Function SetBitScanForward( const Value : LongWord) : Integer;
7157: Function SetBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7158: Function SetBitScanReverse( const Value : LongWord) : Integer;
7159: Function SetBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7160: Function ClearBitScanForward( const Value : LongWord) : Integer;
7161: Function ClearBitScanForward1( const Value, BitIndex : LongWord) : Integer;
7162: Function ClearBitScanReverse( const Value : LongWord) : Integer;
7163: Function ClearBitScanReverse1( const Value, BitIndex : LongWord) : Integer;
7164: Function cReverseBits( const Value : LongWord) : LongWord;
7165: Function cReverseBits1( const Value : LongWord; const BitCount : Integer) : LongWord;
7166: Function cSwapEndian( const Value : LongWord) : LongWord
7167: Function cTwosComplement( const Value : LongWord) : LongWord
7168: Function RotateLeftBits16( const Value : Word; const Bits : Byte) : Word
7169: Function RotateLeftBits32( const Value : LongWord; const Bits : Byte) : LongWord
7170: Function RotateRightBits16( const Value : Word; const Bits : Byte) : Word
7171: Function RotateRightBits32( const Value : LongWord; const Bits : Byte) : LongWord
7172: Function cBitCount( const Value : LongWord) : LongWord
7173: Function cIsPowerOfTwo( const Value : LongWord) : Boolean
7174: Function LowBitMask( const HighBitIndex : LongWord) : LongWord
7175: Function HighBitMask( const LowBitIndex : LongWord) : LongWord
7176: Function RangeBitMask( const LowBitIndex, HighBitIndex : LongWord) : LongWord
7177: Function SetBitRange( const Value : LongWord; const LowBitIndex, HighBitIndex : LongWord) : LongWord
7178: Function ClearBitRange( const Value : LongWord; const LowBitIndex, HighBitIndex : LongWord) : LongWord
7179: Function ToggleBitRange( const Value : LongWord; const LowBitIndex, HighBitIndex : LongWord) : LongWord
7180: Function IsBitRangeSet( const Value : LongWord; const LowBitIndex, HighBitIndex : LongWord) : Boolean
7181: Function IsBitRangeClear( const Value : LongWord; const LowBitIndex, HighBitIndex : LongWord) : Boolean
7182: // AddTypes('CharSet', 'set of AnsiChar
7183: AddTypes('CharSet', 'set of Char //!!!
7184: AddTypes('AnsiCharSet', 'TCharSet
7185: AddTypes('ByteSet', 'set of Byte
7186: AddTypes('AnsiChar', 'Char
7187: // Function AsCharSet( const C : array of AnsiChar) : CharSet
7188: Function AsByteSet( const C : array of Byte) : ByteSet
7189: Procedure ComplementChar( var C : CharSet; const Ch : Char)
7190: Procedure ClearCharSet( var C : CharSet)
7191: Procedure FillCharSet( var C : CharSet)
7192: Procedure ComplementCharSet( var C : CharSet)
7193: Procedure AssignCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7194: Procedure Union( var DestSet : CharSet; const SourceSet : CharSet)
7195: Procedure Difference( var DestSet : CharSet; const SourceSet : CharSet)
7196: Procedure Intersection( var DestSet : CharSet; const SourceSet : CharSet)
7197: Procedure XORCharSet( var DestSet : CharSet; const SourceSet : CharSet)
7198: Function IsSubSet( const A, B : CharSet) : Boolean
7199: Function IsEqual( const A, B : CharSet) : Boolean
7200: Function IsEmpty( const C : CharSet) : Boolean
7201: Function IsComplete( const C : CharSet) : Boolean
7202: Function cCharCount( const C : CharSet) : Integer
7203: Procedure ConvertCaseInsensitive( var C : CharSet)
7204: Function CaseInsensitiveCharSet( const C : CharSet) : CharSet
7205: Function IntRangeLength( const Low, High : Integer) : Int64
7206: Function IntRangeAdjacent( const Low1, High1, Low2, High2 : Integer) : Boolean
7207: Function IntRangeOverlap( const Low1, High1, Low2, High2 : Integer) : Boolean
7208: Function IntRangeHasElement( const Low, High, Element : Integer) : Boolean
7209: Function IntRangeIncludeElement( var Low, High : Integer; const Element : Integer) : Boolean
7210: Function IntRangeIncludeElementRange( var Low, High : Integer; const LowElement, HighElement : Integer) : Boolean
7211: Function CardRangeLength( const Low, High : Cardinal) : Int64
7212: Function CardRangeAdjacent( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7213: Function CardRangeOverlap( const Low1, High1, Low2, High2 : Cardinal) : Boolean
7214: Function CardRangeHasElement( const Low, High, Element : Cardinal) : Boolean
7215: Function CardRangeIncludeElement( var Low, High : Cardinal; const Element : Cardinal) : Boolean
7216: Function CardRangeIncludeElementRange( var Low, High : Cardinal; const LowElement, HighElement : Cardinal) : Boolean;
7217: AddTypes('UnicodeChar', 'WideChar
7218: Function Compare( const I1, I2 : Boolean) : TCompareResult;
7219: Function Compare1( const I1, I2 : Integer) : TCompareResult;
7220: Function Compare2( const I1, I2 : Int64) : TCompareResult;
7221: Function Compare3( const I1, I2 : Extended) : TCompareResult;
7222: Function CompareA( const I1, I2 : AnsiString) : TCompareResult
7223: Function CompareW( const I1, I2 : WideString) : TCompareResult
7224: Function cSgn( const A : LongInt) : Integer;
7225: Function cSgn1( const A : Int64) : Integer;
7226: Function cSgn2( const A : Extended) : Integer;
7227: AddTypes('TConvertResult', '( convertOK, convertFormatError, convertOverflow )
7228: Function AnsiCharToInt( const A : AnsiChar) : Integer
7229: Function WideCharToInt( const A : WideChar) : Integer
7230: Function CharToInt( const A : Char) : Integer
7231: Function IntToAnsiChar( const A : Integer) : AnsiChar
7232: Function IntToWideChar( const A : Integer) : WideChar
7233: Function IntToChar( const A : Integer) : Char
7234: Function IsHexAnsiChar( const Ch : AnsiChar) : Boolean
7235: Function IsHexWideChar( const Ch : WideChar) : Boolean
7236: Function IsHexChar( const Ch : Char) : Boolean
7237: Function HexAnsiCharToInt( const A : AnsiChar) : Integer
7238: Function HexWideCharToInt( const A : WideChar) : Integer
7239: Function HexCharToInt( const A : Char) : Integer

```



```

7240: Function IntToUpperHexAnsiChar( const A : Integer) : AnsiChar
7241: Function IntToUpperHexWideChar( const A : Integer) : WideChar
7242: Function IntToUpperHexChar( const A : Integer) : Char
7243: Function IntToLowerHexAnsiChar( const A : Integer) : AnsiChar
7244: Function IntToLowerHexWideChar( const A : Integer) : WideChar
7245: Function IntToLowerHexChar( const A : Integer) : Char
7246: Function IntToStringA( const A : Int64) : AnsiString
7247: Function IntToStringW( const A : Int64) : WideString
7248: Function IntToString( const A : Int64) : String
7249: Function UIntToStringA( const A : NativeUInt) : AnsiString
7250: Function UIntToStringW( const A : NativeUInt) : WideString
7251: Function UIntToString( const A : NativeUInt) : String
7252: Function LongWordToStrA( const A : LongWord; const Digits : Integer) : AnsiString
7253: Function LongWordToStrW( const A : LongWord; const Digits : Integer) : WideString
7254: Function LongWordToStrU( const A : LongWord; const Digits : Integer) : UnicodeString
7255: Function LongWordToStr( const A : LongWord; const Digits : Integer) : String
7256: Function LongWordToHexA( const A:LongWord;const Digits:Integer;const UpperCase:Boolean):AnsiString;
7257: Function LongWordToHexW( const A:LongWord;const Digits:Integer;const UpperCase:Boolean):WideString;
7258: Function LongWordToHex( const A : LongWord; const Digits : Integer; const UpperCase:Boolean):String
7259: Function LongWordToOctA( const A : LongWord; const Digits : Integer) : AnsiString
7260: Function LongWordToOctW( const A : LongWord; const Digits : Integer) : WideString
7261: Function LongWordToOct( const A : LongWord; const Digits : Integer) : String
7262: Function LongWordToBinA( const A : LongWord; const Digits : Integer) : AnsiString
7263: Function LongWordToBinW( const A : LongWord; const Digits : Integer) : WideString
7264: Function LongWordToBin( const A : LongWord; const Digits : Integer) : String
7265: Function TryStringToInt64A( const S : AnsiString; out A : Int64) : Boolean
7266: Function TryStringToInt64W( const S : WideString; out A : Int64) : Boolean
7267: Function TryStringToInt64( const S : String; out A : Int64) : Boolean
7268: Function StringToInt64DefA( const S : AnsiString; const Default : Int64) : Int64
7269: Function StringToInt64DefW( const S : WideString; const Default : Int64) : Int64
7270: Function StringToInt64Def( const S : String; const Default : Int64) : Int64
7271: Function StringToInt64A( const S : AnsiString) : Int64
7272: Function StringToInt64W( const S : WideString) : Int64
7273: Function StringToInt64( const S : String) : Int64
7274: Function TryStringToIntA( const S : AnsiString; out A : Integer) : Boolean
7275: Function TryStringToIntW( const S : WideString; out A : Integer) : Boolean
7276: Function TryStringToInt( const S : String; out A : Integer) : Boolean
7277: Function StringToIntDefA( const S : AnsiString; const Default : Integer) : Integer
7278: Function StringToIntDefW( const S : WideString; const Default : Integer) : Integer
7279: Function StringToIntDef( const S : String; const Default : Integer) : Integer
7280: Function StringToIntA( const S : AnsiString) : Integer
7281: Function StringToIntW( const S : WideString) : Integer
7282: Function StringToInt( const S : String) : Integer
7283: Function TryStringToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7284: Function TryStringToLongWordW( const S : WideString; out A : LongWord) : Boolean
7285: Function TryStringToLongWord( const S : String; out A : LongWord) : Boolean
7286: Function StringToLongWordA( const S : AnsiString) : LongWord
7287: Function StringToLongWordW( const S : WideString) : LongWord
7288: Function StringToLongWord( const S : String) : LongWord
7289: Function HexToUIntA( const S : AnsiString) : NativeUInt
7290: Function HexToUIntW( const S : WideString) : NativeUInt
7291: Function HexToUInt( const S : String) : NativeUInt
7292: Function TryHexToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7293: Function TryHexToLongWordW( const S : WideString; out A : LongWord) : Boolean
7294: Function TryHexToLongWord( const S : String; out A : LongWord) : Boolean
7295: Function HexToLongWordA( const S : AnsiString) : LongWord
7296: Function HexToLongWordW( const S : WideString) : LongWord
7297: Function HexToLongWord( const S : String) : LongWord
7298: Function TryOctToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7299: Function TryOctToLongWordW( const S : WideString; out A : LongWord) : Boolean
7300: Function TryOctToLongWord( const S : String; out A : LongWord) : Boolean
7301: Function OctToLongWordA( const S : AnsiString) : LongWord
7302: Function OctToLongWordW( const S : WideString) : LongWord
7303: Function OctToLongWord( const S : String) : LongWord
7304: Function TryBinToLongWordA( const S : AnsiString; out A : LongWord) : Boolean
7305: Function TryBinToLongWordW( const S : WideString; out A : LongWord) : Boolean
7306: Function TryBinToLongWord( const S : String; out A : LongWord) : Boolean
7307: Function BinToLongWordA( const S : AnsiString) : LongWord
7308: Function BinToLongWordW( const S : WideString) : LongWord
7309: Function BinToLongWord( const S : String) : LongWord
7310: Function FloatToStringA( const A : Extended) : AnsiString
7311: Function FloatToStringW( const A : Extended) : WideString
7312: Function FloatToString( const A : Extended) : String
7313: Function TryStringToFloatA( const A : AnsiString; out B : Extended) : Boolean
7314: Function TryStringToFloatW( const A : WideString; out B : Extended) : Boolean
7315: Function TryStringToFloat( const A : String; out B : Extended) : Boolean
7316: Function StringToFloatA( const A : AnsiString) : Extended
7317: Function StringToFloatW( const A : WideString) : Extended
7318: Function StringToFloat( const A : String) : Extended
7319: Function StringToFloatDefA( const A : AnsiString; const Default : Extended) : Extended
7320: Function StringToFloatDefW( const A : WideString; const Default : Extended) : Extended
7321: Function StringToFloatDef( const A : String; const Default : Extended) : Extended
7322: Function EncodeBase64(const S,Alphabet:AnsiString; const Pad:Boolean; const PadMultiple:Integer; const
PadChar: AnsiChar) : AnsiString
7323: Function DecodeBase64( const S, Alphabet : AnsiString; const PadSet : CharSet) : AnsiString
7324: unit uPSI_cFundamentUtils;
7325: Const ('b64_MIMEBase64', 'Str').String('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
7326: Const ('b64_UUENcode', 'String').String('!\"#$%&'()*+,-./0123456789;<=>?@ABCDEFGHIJKLMNPQRSTUVWXYZ[\]^_';
7327: Const ('b64_XXENcode', 'String').String('+0123456789ABCDEFGHIJKLMNPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';

```

```

7328: Const('CCHARSET','String').SetString(b64_XXEncode);
7329: Const('CHEXSET','String').SetString('0123456789ABCDEF
7330: Const('HEXDIGITS','String').SetString('0123456789ABCDEF
7331: StHexDigits : array[0..$F] of Char = '0123456789ABCDEF';
7332: Const('DIGISET','String').SetString('0123456789
7333: Const('LETTERSET','String').SetString('ABCDEFGHIJKLMNOPQRSTUVWXYZ
7334: Const('DIGISET2','TCharSet').SetSet('0123456789
7335: Const('LETTERSET2','TCharSet').SetSet('ABCDEFGHIJKLMNOPQRSTUVWXYZ
7336: Const('HEXSET2','TCharSet').SetSet('0123456789ABCDEF');
7337: Const('NUMBERSET','TCharSet').SetSet('0123456789');
7338: Const('NUMBERS','String').SetString('0123456789');
7339: Const('LETTERS','String').SetString('ABCDEFGHIJKLMNOPQRSTUVWXYZ');
7340: Function CharSetToStr( const C : CharSet ) : AnsiString
7341: Function StrToCharSet( const S : AnsiString ) : CharSet
7342: Function MIMEBase64Decode( const S : AnsiString ) : AnsiString
7343: Function MIMEBase64Encode( const S : AnsiString ) : AnsiString
7344: Function UUDecode( const S : AnsiString ) : AnsiString
7345: Function XXDecode( const S : AnsiString ) : AnsiString
7346: Function BytesToHex( const P : array of Byte; const UpperCase : Boolean ) : AnsiString
7347: Function InterfaceToStrA( const I : IInterface ) : AnsiString
7348: Function InterfaceToStrW( const I : IInterface ) : WideString
7349: Function InterfaceToStr( const I : IInterface ) : String
7350: Function ObjectClassName( const O : TObject ) : String
7351: Function ClassClassName( const C : TClass ) : String
7352: Function ObjectToStr( const O : TObject ) : String
7353: Function ObjectToString( const O : TObject ) : String
7354: Function CharSetToStr( const C : CharSet ) : AnsiString
7355: Function StrToCharSet( const S : AnsiString ) : CharSet
7356: Function HashStrA( const S : AnsiString; const Index : Integer; const Count : Integer; const
  AsciiCaseSensitive : Boolean; const Slots : LongWord ) : LongWord
7357: Function HashStrW( const S : WideString; const Index : Integer; const Count : Integer; const
  AsciiCaseSensitive : Boolean; const Slots : LongWord ) : LongWord
7358: Function HashStr( const S : String; const Index : Integer; const Count : Integer; const
  AsciiCaseSensitive : Boolean; const Slots : LongWord ) : LongWord
7359: Function HashInteger( const I : Integer; const Slots : LongWord ) : LongWord
7360: Function HashLongWord( const I : LongWord; const Slots : LongWord ) : LongWord
7361: Const('Bytes1KB','LongInt').SetInt( 1024);
7362: SIRegister_IInterface(CL);
7363: Procedure SelfTestCFundamentUtils
7364:
7365: Function CreateSchedule : IJclSchedule
7366: Function NullStamp : TTimeStamp
7367: Function CompareTimeStamps( const Stamp1, Stamp2 : TTimeStamp ) : Int64
7368: Function EqualTimeStamps( const Stamp1, Stamp2 : TTimeStamp ) : Boolean
7369: Function IsNullTimeStamp( const Stamp : TTimeStamp ) : Boolean
7370:
7371: procedure SIRegister_uwinplot(CL: TPSPascalCompiler);
7372: begin
7373:   AddTypeS('TFunc', 'function(X : Float) : Float;
7374:   Function InitGraphics( Width, Height : Integer ) : Boolean
7375:   Procedure SetWindow( Canvas : TCanvas; X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean)
7376:   Procedure SetOxScale( Scale : TScale; OxMin, OxMax, OxStep : Float)
7377:   Procedure SetOyScale( Scale : TScale; OyMin, OyMax, OyStep : Float)
7378:   Procedure GetOxScale( var Scale : TScale; var OxMin, OxMax, OxStep : Float)
7379:   Procedure GetOyScale( var Scale : TScale; var OyMin, OyMax, OyStep : Float)
7380:   Procedure SetGraphTitle( Title : String)
7381:   Procedure SetOxTitle( Title : String)
7382:   Procedure SetOyTitle( Title : String)
7383:   Function GetGraphTitle : String
7384:   Function GetOxTitle : String
7385:   Function GetOyTitle : String
7386:   Procedure PlotOxAxis( Canvas : TCanvas)
7387:   Procedure PlotOyAxis( Canvas : TCanvas)
7388:   Procedure PlotGrid( Canvas : TCanvas; Grid : TGrid)
7389:   Procedure WriteGraphTitle( Canvas : TCanvas)
7390:   Function SetMaxCurv( NCurv : Byte ) : Boolean
7391:   Procedure SetPointParam( CurvIndex, Symbol, Size : Integer; Color : TColor)
7392:   Procedure SetLineParam( CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor)
7393:   Procedure SetCurvLegend( CurvIndex : Integer; Legend : String)
7394:   Procedure SetCurvStep( CurvIndex, Step : Integer)
7395:   Function GetMaxCurv : Byte
7396:   Procedure GetPointParam( CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor)
7397:   Procedure GetLineParam( CurvIndex:Integer;var Style:TPenStyle;var Width:Integer; var Color:TColor);
7398:   Function GetCurvLegend( CurvIndex : Integer ) : String
7399:   Function GetCurvStep( CurvIndex : Integer ) : Integer
7400:   Procedure PlotPoint( Canvas : TCanvas; X, Y : Float; CurvIndex : Integer)
7401:   Procedure PlotCurve( Canvas : TCanvas; X, Y : TVector; Lb, Ub, CurvIndex : Integer)
7402:   Procedure PlotCurveWithErrorBars(Canvas : TCanvas; X,Y,S: TVector; Ns,Lb,Ub,CurvIndex:Integer)
7403:   Procedure PlotFunc(Canvas: TCanvas; Func: TFunc; Xmin,Xmax: Float; Npt,CurvIndex: Integer)
7404:   Procedure WriteLegend( Canvas : TCanvas; NCurv : Integer; ShowPoints, ShowLines : Boolean)
7405:   Procedure ConRec( Canvas : TCanvas; Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix)
7406:   Function Xpixel( X : Float ) : Integer
7407:   Function Ypixel( Y : Float ) : Integer
7408:   Function Xuser( X : Integer ) : Float
7409:   Function Yuser( Y : Integer ) : Float
7410: end;
7411:
7412: Procedure FFT( NumSamples : Integer; InArray, OutArray : TCompVector)
7413: Procedure IFFT( NumSamples : Integer; InArray, OutArray : TCompVector)

```

```

7414: Procedure FFT_Integer( NumSamples : Integer; RealIn, ImagIn : TIntVector; OutArray : TCompVector)
7415: Procedure FFT_Integer_Cleanup
7416: Procedure CalcFrequency(NumSamples,FrequencyIndex: Integer;InArray: TCompVector;var FT : Complex)
7417: //unit uPSI_JclStreams;
7418: Function StreamSeek( Stream : TStream; const Offset : Int64; const Origin : TSeekOrigin) : Int64
7419: Function StreamCopy( Source : TStream; Dest : TStream; BufferSize : Integer) : Int64
7420: Function CompareStreams( A, B : TStream; BufferSize : Integer) : Boolean
7421: Function JCompareFiles( const FileA, FileB : TFileName; BufferSize : Integer) : Boolean
7422:
7423: procedure SIRegister_FmxUtils(CL: TPSPascalCompiler);
7424: begin
7425:   FindClass('TOBJECT'),'EInvalidDest
7426:   FindClass('TOBJECT'),'EFCantMove
7427:   Procedure fmxCopyFile( const FileName, DestName : string)
7428:   Procedure fmxMoveFile( const FileName, DestName : string)
7429:   Function fmxGetFileSize( const FileName : string) : LongInt
7430:   Function fmxFileDateTime( const FileName : string) : TDateTime
7431:   Function fmxHasAttr( const FileName : string; Attr : Word) : Boolean
7432:   Function fmxExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer):THandle;
7433: end;
7434:
7435: procedure SIRegister_FindFileIter(CL: TPSPascalCompiler);
7436: begin
7437:   SIRegister_IFindFileIterator(CL);
7438:   Function CreateFindFile(const Path:string; IncludeAttr:Integer;out iffi:IFindFileIterator):Bool;
7439: end;
7440:
7441: procedure SIRegister_PCharUtils(CL: TPSPascalCompiler);
7442: begin
7443:   Function SkipWhite( cp : PChar) : PChar
7444:   Function ReadStringDoubleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7445:   Function ReadStringSingleQuotedMaybe( cp : PChar; var AStr : string) : PChar
7446:   Function ReadIdent( cp : PChar; var ident : string) : PChar
7447: end;
7448:
7449: procedure SIRegister_JclStrHashMap(CL: TPSPascalCompiler);
7450: begin
7451:   SIRegister_TStringHashMapTraits(CL);
7452:   Function CaseSensitiveTraits : TStringHashMapTraits
7453:   Function CaseInsensitiveTraits : TStringHashMapTraits
7454:   THashNode', 'record Str : string; Ptr : Pointer; Left : PHashNod'
7455:   + 'e; Right : PHashNode; end
7456:   //PHashArray', '^THashArray // will not work
7457:   SIRegister_TStringHashMap(CL);
7458:   THashValue', 'Cardinal
7459:   Function StrHash( const s : string) : THashValue
7460:   Function TextHash( const s : string) : THashValue
7461:   Function DataHash( var AValue, ASize : Cardinal) : THashValue
7462:   Function Iterate_FreeObjects( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7463:   Function Iterate_Dispose( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7464:   Function Iterate_FreeMem( AUserData : Pointer; const AStr : string; var AData : Pointer) : Boolean
7465:   SIRegister_TCaseSensitiveTraits(CL);
7466:   SIRegister_TCaseInsensitiveTraits(CL);
7467:
7468:
7469: //*****unit uPSI_umath;
7470: Function uExpo( X : Float) : Float
7471: Function uExp2( X : Float) : Float
7472: Function uExp10( X : Float) : Float
7473: Function uLog( X : Float) : Float
7474: Function uLog2( X : Float) : Float
7475: Function uLog10( X : Float) : Float
7476: Function uLogA( X, A : Float) : Float
7477: Function uIntPower( X : Float; N : Integer) : Float
7478: Function uPower( X, Y : Float) : Float
7479: Function SgnGamma( X : Float) : Integer
7480: Function Stirling( X : Float) : Float
7481: Function StirLog( X : Float) : Float
7482: Function Gamma( X : Float) : Float
7483: Function LnGamma( X : Float) : Float
7484: Function DiGamma( X : Float) : Float
7485: Function TriGamma( X : Float) : Float
7486: Function IGamma( X : Float) : Float
7487: Function JGamma( X : Float) : Float
7488: Function InvGamma( X : Float) : Float
7489: Function Erf( X : Float) : Float
7490: Function Erfc( X : Float) : Float
7491: Function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7492: { Correlation coefficient between samples X and Y }
7493: function DBeta(A, B, X : Float) : Float;
7494: { Density of Beta distribution with parameters A and B }
7495: Function LambertW( X : Float; UBranch, Offset : Boolean) : Float
7496: Function Beta(X, Y : Float) : Float
7497: Function Binomial( N, K : Integer) : Float
7498: Function PBinom( N : Integer; P : Float; K : Integer) : Float
7499: Procedure Cholesky( A, L : TMatrix; Lb, Ub : Integer)
7500: Procedure LU_Decom( A : TMatrix; Lb, Ub : Integer)
7501: Procedure LU_Solve( A : TMatrix; B : TVector; Lb, Ub : Integer; X : TVector)
7502: Function DNorm( X : Float) : Float

```

```

7503:
7504: function DGamma(A, B, X : Float) : Float;
7505: { Density of Gamma distribution with parameters A and B }
7506: function DKhi2(Nu : Integer; X : Float) : Float;
7507: { Density of Khi-2 distribution with Nu d.o.f. }
7508: function DStudent(Nu : Integer; X : Float) : Float;
7509: { Density of Student distribution with Nu d.o.f. }
7510: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float;
7511: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
7512: function IBeta(A, B, X : Float) : Float;
7513: { Incomplete Beta function}
7514: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float;
7515:
7516: procedure SIRegister_unlfit(CL: TPSPascalCompiler);
7517: begin
7518:   procedure SetOptAlgo( Algo : TOptAlgo)
7519: procedure SetOptAlgo(Algo : TOptAlgo);
7520: { -----
7521:   Sets the optimization algorithm according to Algo, which must be
7522:   NL_MARQ, NL_SIMP, NL_BFGS, NL_SA, NL_GA. Default is NL_MARQ   }
7523:
7524:   Function GetOptAlgo : TOptAlgo
7525:   Procedure SetMaxParam( N : Byte)
7526:   Function GetMaxParam : Byte
7527:   Procedure SetParamBounds( I : Byte; ParamMin, ParamMax : Float)
7528:   Procedure GetParamBounds( I : Byte; var ParamMin, ParamMax : Float)
7529:   Function NullParam( B : TVector; Lb, Ub : Integer) : Boolean
7530:   Procedure NLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y : TVector; Lb, Ub : Integer; MaxIter :
Integer; Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7531:   Procedure WNLFit( RegFunc : TRegFunc; DerivProc : TDerivProc; X, Y, S : TVector; Lb, Ub:Integer;
MaxIter:Integer;Tol : Float; B : TVector; FirstPar, LastPar : Integer; V : TMatrix)
7532:   Procedure SetMCFile( FileName : String)
7533:   Procedure SimFit(RegFunc:TRegFunc;X,Y:TVector;Lb,Ub:Integer;B:TVector;FirstPar,LastPar:Integer;V:TMatrix;
7534:   Procedure WSimFit(RegFunc:TRegFunc; X,Y,S:TVector;Lb,Ub:Integer;B:TVector;FirstPar,
LastPar:Integer;V:TMatrix);
7535: end;
7536:
7537: (*-----*)
7538: procedure SIRegister_usimplex(CL: TPSPascalCompiler);
7539: begin
7540:   Procedure SaveSimplex( FileName : string)
7541:   Procedure Simplex(Func:TFuncNVar; X:TVector;Lb,Ub:Integer; MaxIter:Integer;Tol:Float; var F_min:Float);
7542: end;
7543: (*-----*)
7544: procedure SIRegister_uregtest(CL: TPSPascalCompiler);
7545: begin
7546:   Procedure RegTest(Y,Ycalc: TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest)
7547:   Procedure WRegTest(Y,Ycalc,S:TVector;LbY,UbY:Integer;V:TMatrix;LbV,UbV:Integer;var Test:TRegTest);
7548: end;
7549:
7550: procedure SIRegister_ustrings(CL: TPSPascalCompiler);
7551: begin
7552:   Function LTrim( S : String) : String
7553:   Function RTrim( S : String) : String
7554:   Function UTrim( S : String) : String
7555:   Function StrChar( N : Byte; C : Char) : String
7556:   Function RFill( S : String; L : Byte) : String
7557:   Function LFill( S : String; L : Byte) : String
7558:   Function CFill( S : String; L : Byte) : String
7559:   Function Replace( S : String; C1, C2 : Char) : String
7560:   Function Extract( S : String; var Index : Byte; Delim : Char) : String
7561:   Procedure Parse( S : String; Delim : Char; Field : TStrVector; var N : Byte)
7562:   Procedure SetFormat( NumLength, MaxDec : Integer; FloatPoint, NSZero : Boolean)
7563:   Function FloatStr( X : Float) : String
7564:   Function IntStr( N : LongInt) : String
7565:   Function uCompStr( Z : Complex) : String
7566: end;
7567:
7568: procedure SIRegister_uhyper(CL: TPSPascalCompiler);
7569: begin
7570:   Function uSinh( X : Float) : Float
7571:   Function uCosh( X : Float) : Float
7572:   Function uTanh( X : Float) : Float
7573:   Function uArcSinh( X : Float) : Float
7574:   Function uArcCosh( X : Float) : Float
7575:   Function ArcTanh( X : Float) : Float
7576:   Procedure SinhCosh( X : Float; var SinhX, CoshX : Float)
7577: end;
7578:
7579: procedure SIRegister_urandom(CL: TPSPascalCompiler);
7580: begin
7581:   type RNG_Type =
7582:     (RNG_MWC,      { Multiply-With-Carry }
7583:     RNG_MT,        { Mersenne Twister }
7584:     RNG_UVAG);     { Universal Virtual Array Generator }
7585:   Procedure SetRNG( RNG : RNG_Type)
7586:   Procedure InitGen( Seed : RNG_IntType)
7587:   Procedure SRand( Seed : RNG_IntType)
7588:   Function IRandGen : RNG_IntType

```



```

7589: Function IRanGen31 : RNG_IntType
7590: Function RanGen1 : Float
7591: Function RanGen2 : Float
7592: Function RanGen3 : Float
7593: Function RanGen53 : Float
7594: end;
7595:
7596: // Optimization by Simulated Annealing
7597: procedure SIRegister_usimann(CL: TPSPascalCompiler);
7598: begin
7599:   Procedure InitSAParams( NT, NS, NCycles : Integer; RT : Float)
7600:   Procedure SA_CreateLogFile( FileName : String)
7601:   Procedure SimAnn(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7602: end;
7603:
7604: procedure SIRegister_uranuvag(CL: TPSPascalCompiler);
7605: begin
7606:   Procedure InitUVAGbyString( KeyPhrase : string)
7607:   Procedure InitUVAG( Seed : RNG_IntType)
7608:   Function IRanUVAG : RNG_IntType
7609: end;
7610:
7611: procedure SIRegister_uenalg(CL: TPSPascalCompiler);
7612: begin
7613:   Procedure InitGAParams( NP, NG : Integer; SR, MR, HR : Float)
7614:   Procedure GA_CreateLogFile( LogFileName : String)
7615:   Procedure GenAlg(Func: TFuncNVar; X, Xmin, Xmax : TVector; Lb, Ub : Integer; var F_min : Float);
7616: end;
7617:
7618: TVector', 'array of Float
7619: procedure SIRegister_uqsort(CL: TPSPascalCompiler);
7620: begin
7621:   Procedure QSort( X : TVector; Lb, Ub : Integer)
7622:   Procedure DQSort( X : TVector; Lb, Ub : Integer)
7623: end;
7624:
7625: procedure SIRegister_uinterv(CL: TPSPascalCompiler);
7626: begin
7627:   Procedure Interval( X1, X2 : Float; MinDiv, MaxDiv : Integer; var Min, Max, Step : Float)
7628:   Procedure AutoScale(X: TVector; Lb, Ub : Integer; Scale : TScale; var XMin, XMax, XStep:Float)
7629: end;
7630:
7631: procedure SIRegister_D2XXUnit(CL: TPSPascalCompiler);
7632: begin
7633:   FT_Result', 'Integer
7634:   //TDWordptr', '^DWord // will not work
7635:   TFT_Program_Data', record Signature1 : DWord; Signature2 : DWor
7636:   d; Version : DWord; VendorID : Word; ProductID : Word; Manufacturer : PCha
7637:   r; ManufacturerID : PChar; Description : PChar; SerialNumber : PChar; MaxP
7638:   ower : Word; PnP : Word; SelfPowered : Word; RemoteWakeUp : Word; Rev4 : B
7639:   yte; IsoIn : Byte; IsoOut : Byte; PullDownEnable : Byte; SerNumEnable : By
7640:   te; USBVersionEnable : Byte; USBVersion : Word; Rev5 : Byte; IsoInA : Byte
7641:   ; IsoInB : Byte; IsoOutA : Byte; IsoOutB : Byte; PullDownEnable5 : Byte; S
7642:   erNumEnable5 : Byte; USBVersionEnable5 : Byte; USBVersion5 : Word; AIsHigh
7643:   Current : Byte; BIsHighCurrent : Byte; IFAIsFifo : Byte; IFAIsFifoTar : By
7644:   te; IFAIsFastSer : Byte; AIsVCP : Byte; IFBIsFifo : Byte; IFBIsFifoTar : B
7645:   yte; IFBIsFastSer : Byte; BIsVCP : Byte; UseExtOsc : Byte; HighDriveIOs :
7646:   Byte; EndpointSize : Byte; PullDownEnableR : Byte; SerNumEnableR : Byte; I
7647:   nvertTXD : Byte; InvertRXD : Byte; InvertRTS : Byte; InvertCTS : Byte; Inv
7648:   ertDTR : Byte; InvertDSR : Byte; InvertDCD : Byte; InvertRI : Byte; Cbus0
7649:   : Byte; Cbus1 : Byte; Cbus2 : Byte; Cbus3 : Byte; Cbus4 : Byte; RIsVCP : B
7650:   yte; end
7651: end;
7652:
7653:
7654: ***** PaintFX*****
7655: procedure SIRegister_TJvPaintFX(CL: TPSPascalCompiler);
7656: begin
7657:   //with RegClassS(CL, 'TComponent', 'TJvPaintFX') do
7658:   with AddClassN(FindClass('TComponent'), 'TJvPaintFX') do begin
7659:     Procedure Solarize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7660:     Procedure Posterize( const Src : TBitmap; var Dst : TBitmap; Amount : Integer)
7661:     Procedure Blend( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7662:     Procedure Blend2( const Src1, Src2 : TBitmap; var Dst : TBitmap; Amount : Single)
7663:     Procedure ExtractColor( const Dst : TBitmap; AColor : TColor)
7664:     Procedure ExcludeColor( const Dst : TBitmap; AColor : TColor)
7665:     Procedure Turn( Src, Dst : TBitmap)
7666:     Procedure TurnRight( Src, Dst : TBitmap)
7667:     Procedure HeightMap( const Dst : TBitmap; Amount : Integer)
7668:     Procedure TexturizeTile( const Dst : TBitmap; Amount : Integer)
7669:     Procedure TexturizeOverlap( const Dst : TBitmap; Amount : Integer)
7670:     Procedure RippleRandom( const Dst : TBitmap; Amount : Integer)
7671:     Procedure RippleTooth( const Dst : TBitmap; Amount : Integer)
7672:     Procedure RippleTriangle( const Dst : TBitmap; Amount : Integer)
7673:     Procedure Triangles( const Dst : TBitmap; Amount : Integer)
7674:     Procedure DrawMandelJulia(const Dst: TBitmap; x0,y0,x1,y1:Single; Niter:Integer; Mandel:Boolean)
7675:     Procedure FilterXBlue( const Dst : TBitmap; Min, Max : Integer)
7676:     Procedure FilterXGreen( const Dst : TBitmap; Min, Max : Integer)
7677:     Procedure FilterXRed( const Dst : TBitmap; Min, Max : Integer)

```

```

7678: Procedure FilterBlue( const Dst : TBitmap; Min, Max : Integer)
7679: Procedure FilterGreen( const Dst : TBitmap; Min, Max : Integer)
7680: Procedure FilterRed( const Dst : TBitmap; Min, Max : Integer)
7681: Procedure Emboss( var Bmp : TBitmap)
7682: Procedure Plasma( Src1, Src2, Dst : TBitmap; Scale, Turbulence : Single)
7683: Procedure Shake( Src, Dst : TBitmap; Factor : Single)
7684: Procedure ShakeDown( Src, Dst : TBitmap; Factor : Single)
7685: Procedure KeepBlue( const Dst : TBitmap; Factor : Single)
7686: Procedure KeepGreen( const Dst : TBitmap; Factor : Single)
7687: Procedure KeepRed( const Dst : TBitmap; Factor : Single)
7688: Procedure Mandelbrot( const Dst : TBitmap; Factor : Integer)
7689: Procedure MaskMandelbrot( const Dst : TBitmap; Factor : Integer)
7690: Procedure FoldRight( Src1, Src2, Dst : TBitmap; Amount : Single)
7691: Procedure QuartoOpaque( Src, Dst : TBitmap)
7692: Procedure SemiOpaque( Src, Dst : TBitmap)
7693: Procedure ShadowDownLeft( const Dst : TBitmap)
7694: Procedure ShadowDownRight( const Dst : TBitmap)
7695: Procedure ShadowUpLeft( const Dst : TBitmap)
7696: Procedure ShadowUpRight( const Dst : TBitmap)
7697: Procedure Darkness( const Dst : TBitmap; Amount : Integer)
7698: Procedure Trace( const Dst : TBitmap; Intensity : Integer)
7699: Procedure FlipRight( const Dst : TBitmap)
7700: Procedure FlipDown( const Dst : TBitmap)
7701: Procedure SpotLight( const Dst : TBitmap; Amount : Integer; Spot : TRect)
7702: Procedure SplitLight( const Dst : TBitmap; Amount : Integer)
7703: Procedure MakeSeamlessClip( var Dst : TBitmap; Seam : Integer)
7704: Procedure Wave( const Dst : TBitmap; Amount, Inference, Style : Integer)
7705: Procedure Mosaic( const Bm : TBitmap; Size : Integer)
7706: Procedure SmoothRotate( var Src, Dst : TBitmap; CX, CY : Integer; Angle : Single)
7707: Procedure SmoothResize( var Src, Dst : TBitmap)
7708: Procedure Twist( var Bmp, Dst : TBitmap; Amount : Integer)
7709: Procedure SplitBlur( const Dst : TBitmap; Amount : Integer)
7710: Procedure GaussianBlur( const Dst : TBitmap; Amount : Integer)
7711: Procedure Smooth( const Dst : TBitmap; Weight : Integer)
7712: Procedure GrayScale( const Dst : TBitmap)
7713: Procedure AddColorNoise( const Dst : TBitmap; Amount : Integer)
7714: Procedure AddMonoNoise( const Dst : TBitmap; Amount : Integer)
7715: Procedure Contrast( const Dst : TBitmap; Amount : Integer)
7716: Procedure Lightness( const Dst : TBitmap; Amount : Integer)
7717: Procedure Saturation( const Dst : TBitmap; Amount : Integer)
7718: Procedure Spray( const Dst : TBitmap; Amount : Integer)
7719: Procedure AntiAlias( const Dst : TBitmap)
7720: Procedure AntiAliasRect( const Dst : TBitmap; XOrigin, YOrigin, XFinal, YFinal : Integer)
7721: Procedure SmoothPoint( const Dst : TBitmap; XK, YK : Integer)
7722: Procedure FishEye( var Bmp, Dst : TBitmap; Amount : Single)
7723: Procedure Marble( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7724: Procedure Marble2( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7725: Procedure Marble3( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7726: Procedure Marble4( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7727: Procedure Marble5( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7728: Procedure Marble6( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7729: Procedure Marble7( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7730: Procedure Marble8( const Src : TBitmap; var Dst : TBitmap; Scale : Single; Turbulence : Integer)
7731: Procedure SqueezeHor( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7732: Procedure SplitRound( Src, Dst : TBitmap; Amount : Integer; Style : TLightBrush)
7733: Procedure Tile( Src, Dst : TBitmap; Amount : Integer)
7734: Procedure Stretch( Src, Dst : TBitmap; Filter : TFilterProc; AWidth : Single)
7735: Procedure Grow( Src1, Src2, Dst : TBitmap; Amount : Single; X, Y : Integer)
7736: Procedure Invert( Src : TBitmap)
7737: Procedure MirrorRight( Src : TBitmap)
7738: Procedure MirrorDown( Src : TBitmap)
7739: end;
7740: end;
7741:
7742: (*-----*)
7743: procedure SIRegister_JvPaintFX(CL: TPSPascalCompiler);
7744: begin
7745:   AddTypeS('TLightBrush', '( lbBrightness, lbContrast, lbSaturation, lbFishe'
7746:     + 'ye, lbrotate, lbtwist, lbripple, mbHor, mbTop, mbBottom, mbDiamond, mbWast'
7747:     + 'e, mbRound, mbRound2, mbSplitRound, mbSplitWaste )'
7748:   SIRegister_TJvPaintFX(CL);
7749:   Function SplineFilter( Value : Single) : Single
7750:   Function BellFilter( Value : Single) : Single
7751:   Function TriangleFilter( Value : Single) : Single
7752:   Function BoxFilter( Value : Single) : Single
7753:   Function HermiteFilter( Value : Single) : Single
7754:   Function Lanczos3Filter( Value : Single) : Single
7755:   Function MitchellFilter( Value : Single) : Single
7756: end;
7757:
7758:
7759: (*-----*)
7760: procedure SIRegister_Chart(CL: TPSPascalCompiler);
7761: begin
7762:   'TeeMsg_DefaultFunctionName', 'String').SetString( 'TeeFunction
7763:   TeeMsg_DefaultSeriesName', 'String').SetString( 'Series
7764:   TeeMsg_DefaultToolName', 'String').SetString( 'ChartTool
7765:   ChartComponentPalette', 'String').SetString( 'TeeChart
7766:   TeeMaxLegendColumns', 'LongInt').SetInt( 2);

```

```

7767: TeeDefaultLegendSymbolWidth', 'LongInt').SetInt( 20);
7768: TeeTitleFootDistance, LongInt).SetInt( 5);
7769: SIRegister_TCustomChartWall(CL);
7770: SIRegister_TChartWall(CL);
7771: SIRegister_TChartLegendGradient(CL);
7772: TLegendStyle', '( lsAuto, lsSeries, lsValues, lsLastValues, lsSeriesGroups )
7773: TLegendAlignment', '( laLeft, laRight, laTop, laBottom )
7774: FindClass('TObject'), 'LegendException
7775: TOnGetLegendText', 'Procedure ( Sender : TCustomAxisPanel; Legen'
7776: + 'dStyle : TLegendStyle; Index : Integer; var LegendText : String)
7777: FindClass('TObject'), 'TCustomChartLegend
7778: TLegendSymbolSize', '( lcsPercent, lcsPixels )
7779: TLegendSymbolPosition', '( spLeft, spRight )
7780: TSymbolDrawEvent', 'Procedure(Sender:TObject;Series:TChartSeries;ValueIndex:Integer;R:TRect);
7781: TSymbolCalcHeight', 'Function : Integer
7782: SIRegister_TLegendSymbol(CL);
7783: SIRegister_TTeeCustomShapePosition(CL);
7784: TCheckBoxesStyle', '( cbsCheck, cbsRadio )
7785: SIRegister_TLegendTitle(CL);
7786: SIRegister_TLegendItem(CL);
7787: SIRegister_TLegendItems(CL);
7788: TLegendCalcSize', 'Procedure ( Sender : TCustomChartLegend; var ASize : Integer)
7789: FindClass('TObject'), 'TCustomChart
7790: SIRegister_TCustomChartLegend(CL);
7791: SIRegister_TChartLegend(CL);
7792: SIRegister_TChartTitle(CL);
7793: SIRegister_TChartFootTitle(CL);
7794: TChartClick', 'Procedure ( Sender : TCustomChart; Button : TMous'
7795: + 'eButton; Shift : TShiftState; X, Y : Integer)
7796: TChartClickAxis', 'Procedure ( Sender : TCustomChart; Axis : TCh'
7797: + 'artAxis; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7798: TChartClickSeries', 'Procedure ( Sender : TCustomChart; Series : '
7799: + 'TChartSeries; ValueIndex : Integer; Button : TMouseButton; Shift:TShiftState;X,Y:Integer)
7800: TChartClickTitle', 'Procedure ( Sender : TCustomChart; ATitle : '
7801: + 'TChartTitle; Button : TMouseButton; Shift : TShiftState; X, Y : Integer)
7802: TOnGetLegendPos', 'Procedure (Sender: TCustomChart; Index: Integer; var X,Y,XColor:Integer)
7803: TOnGetLegendRect', 'Procedure ( Sender : TCustomChart; var Rect : TRect)
7804: TAxisSavedScales', 'record Auto : Boolean; AutoMin : Boolean; Au'
7805: + 'toMax : Boolean; Min : Double; Max : Double; end
7806: TAllAxisSavedScales', 'array of TAxisSavedScales
7807: SIRegister_TChartBackWall(CL);
7808: SIRegister_TChartRightWall(CL);
7809: SIRegister_TChartBottomWall(CL);
7810: SIRegister_TChartLeftWall(CL);
7811: SIRegister_TChartWalls(CL);
7812: TChartAllowScrollEvent', 'Procedure(Sender:TChartAxis;var AMin,AMax:Double;var AllowScroll:Boolean);
7813: SIRegister_TCustomChart(CL);
7814: SIRegister_TChart(CL);
7815: SIRegister_TTeeSeriesTypes(CL);
7816: SIRegister_TTeeToolTypes(CL);
7817: SIRegister_TTeeDragObject(CL);
7818: SIRegister_TColorPalettes(CL);
7819: Procedure RegisterTeeSeries(ASeriesClass:TChartSeriesClass;ADescription,
AGalleryPage:PString;ANumGallerySeries:Integer;
7820: Procedure RegisterTeeSeries1( ASeriesClass : TChartSeriesClass; ADescription : PString);
7821: Procedure RegisterTeeFunction(AFunctClass:TeeFunctionClass;ADescription,
AGalleryPage:PString;ANumGallerySeries: Int;
7822: Procedure RegisterTeeBasicFunction( AFunctionClass : TTeeFunctionClass; ADescription : PString)
7823: Procedure RegisterTeeSeriesFunction(ASeriesClass: TChartSeriesClass;AFunctionClass:TTeeFunctionClass;
ADescription, AGalleryPage : PString; ANumGallerySeries : Integer; ASubIndex : Integer)
7824: Procedure UnRegisterTeeSeries( const ASeriesList : array of TChartSeriesClass)
7825: Procedure UnRegisterTeeFunctions( const AFunctionList : array of TTeeFunctionClass)
7826: Procedure AssignSeries( var OldSeries, NewSeries : TChartSeries)
7827: Function CreateNewTeeFunction( ASeries : TChartSeries; AClass : TTeeFunctionClass) : TTeeFunction
7828: Function CreateNewSeries( AOwner : TComponent; AChart : TCustomAxisPanel; AClass : TChartSeriesClass;
AFunctionClass : TTeeFunctionClass) : TChartSeries
7829: Function CloneChartSeries( ASeries : TChartSeries) : TChartSeries;
7830: Function CloneChartSeries1( ASeries : TChartSeries; AChart : TCustomAxisPanel) : TChartSeries;
7831: Function CloneChartSeries2(ASeries:TChartSeries;AOwner:TComponent;AChart:TCustomAxisPanel):TChartSeries;;
7832: Function CloneChartTool( ATool : TTeeCustomTool; AOwner : TComponent) : TTeeCustomTool
7833: Function ChangeSeriesType( var ASeries : TChartSeries; NewType : TChartSeriesClass) : TChartSeries
7834: Procedure ChangeAllSeriesType( AChart : TCustomChart; AClass : TChartSeriesClass)
7835: Function GetNewSeriesName( AOwner : TComponent) : TComponentName
7836: Procedure RegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7837: Procedure UnRegisterTeeTools( const ATools : array of TTeeCustomToolClass)
7838: Function GetGallerySeriesName( ASeries : TChartSeries) : String
7839: Procedure PaintSeriesLegend(ASeries:TChartSeries; ACanvas:TCanvas; const R:TRect;ReferenceChart :
TCustomChart);
7840: SIRegister_TChartTheme(CL);
7841: //TChartThemeClass', 'class of TChartTheme
7842: //TCanvasClass', 'class of TCanvas3D
7843: Function SeriesNameOrIndex( ASeries : TCustomChartSeries) : String
7844: Function SeriesTitleOrName( ASeries : TCustomChartSeries) : String
7845: Procedure FillSeriesItems( AItems : TStrings; AList : TCustomSeriesList; UseTitles : Boolean)
7846: Procedure ShowMessageUser( const S : String)
7847: Function HasNoMandatoryValues( ASeries : TChartSeries) : Boolean
7848: Function HasLabels( ASeries : TChartSeries) : Boolean
7849: Function HasColors( ASeries : TChartSeries) : Boolean
7850: Function SeriesGuessContents( ASeries : TChartSeries) : TeeFormatFlag

```

```

7851: Procedure TeeDrawBitmapEditor( Canvas : TCanvas; Element : TCustomChartElement; Left, Top : Integer)
7852: end;
7853:
7854:
7855: procedure SIRegister_TeeProcs(CL: TPSPascalCompiler);
7856: begin
7857:   /'TeeFormBorderStyle', '').SetString( bsNone);
7858:   SIRegister_TMetafile(CL);
7859:   'TeeDefVerticalMargin', 'LongInt').SetInt( 4);
7860:   'TeeDefHorizMargin', 'LongInt').SetInt( 3);
7861:   'crTeeHand', 'LongInt').SetInt( TCursor ( 2020 ));
7862:   'TeeMsg_TeeHand', 'String').SetString( 'crTeeHand
7863:   'TeeNormalPrintDetail', 'LongInt').SetInt( 0);
7864:   'TeeHighPrintDetail', 'LongInt').SetInt( - 100);
7865:   'TeeDefault_PrintMargin', 'LongInt').SetInt( 15);
7866:   'MaxDefaultColors', 'LongInt').SetInt( 19);
7867:   'TeeTabDelimiter', 'Char').SetString( #9);
7868:   TDateTimeStep', '( dtOneMicroSecond, dtOneMillisecond, dtOneSeco'
7869:   + 'nd, dtFiveSeconds, dtTenSeconds, dtFifteenSeconds, dtThirtySeconds, dtOneM'
7870:   + 'inute, dtFiveMinutes, dtTenMinutes, dtFifteenMinutes, dtThirtyMinutes, dtO'
7871:   + 'neHour, dtTwoHours, dtSixHours, dtTwelveHours, dtOneDay, dtTwoDays, dtThre'
7872:   + 'eDays, dtOneWeek, dtHalfMonth, dtOneMonth, dtTwoMonths, dtThreeMonths, dtF'
7873:   + 'ourMonths, dtSixMonths, dtOneYear, dtNone )
7874:   SIRegister_TCustomPanelNoCaption(CL);
7875:   FindClass('TObJECT'), 'TCustomTeePanel
7876:   SIRegister_TZoomPanning(CL);
7877:   SIRegister_TTeeEvent(CL);
7878:   //SIRegister_TTeeEventListeners(CL);
7879:   TTeeMouseEventKind', '( meDown, meUp, meMove )
7880:   SIRegister_TTeeMouseEvent(CL);
7881:   SIRegister_TCustomTeePanel(CL);
7882:   //TChartGradient', 'TGradient
7883:   //TChartGradientClass', 'class of TChartGradient
7884:   TPanningMode', '( pmNone, pmHorizontal, pmVertical, pmBoth )
7885:   SIRegister_TTeeZoomPen(CL);
7886:   SIRegister_TTeeZoomBrush(CL);
7887:   TTeeZoomDirection', '( tzdHorizontal, tzdVertical, tzdBoth )
7888:   SIRegister_TTeeZoom(CL);
7889:   FindClass('TObJECT'), 'TCustomTeePanelExtended
7890:   TTeeBackImageMode', '( pbmStretch, pbmTile, pbmCenter, pbmCustom )
7891:   SIRegister_TBBackImage(CL);
7892:   SIRegister_TCustomTeePanelExtended(CL);
7893:   //TChartBrushClass', 'class of TChartBrush
7894:   SIRegister_TTeeCustomShapeBrushPen(CL);
7895:   TChartObjectShapeStyle', '( fosRectangle, fosRoundRectangle, fosEllipse )
7896:   TTextFormat', '( ttfNormal, ttfHtml )
7897:   SIRegister_TTeeCustomShape(CL);
7898:   SIRegister_TTeeShape(CL);
7899:   SIRegister_TTeeExportData(CL);
7900:   Function TeeStr( const Num : Integer) : String
7901:   Function DateTimeDefaultFormat( const AStep : Double) : String
7902:   Function TEEDaysInMonth( Year, Month : Word) : Word
7903:   Function FindDateTimeStep( const StepValue : Double) : TDateTimeStep
7904:   Function NextDateTimeStep( const AStep : Double) : Double
7905:   Function PointInLine( const P : TPoint; const px, py, qx, qy : Integer) : Boolean;
7906:   Function PointInLine1( const P, FromPoint, ToPoint : TPoint) : Boolean;
7907:   Function PointInLine2(const P,FromPoint,ToPoint:TPoint;const TolerancePixels:Integer):Boolean;
7908:   Function PointInLine3( const P : TPoint; const px, py, qx, qy, TolerancePixels:Integer):Boolean;
7909:   Function PointInLineTolerance(const P:TPoint;const px,py,qx,qy,TolerancePixels:Integer):Boolean;
7910:   Function PointInPolygon( const P : TPoint; const Poly : array of TPoint) : Boolean
7911:   Function PointInTriangle( const P, P0, P1, P2 : TPoint) : Boolean;
7912:   Function PointInTriangle1( const P : TPoint; X0, X1, Y0, Y1 : Integer) : Boolean;
7913:   Function PointInHorizTriangle( const P : TPoint; Y0, Y1, X0, X1 : Integer) : Boolean
7914:   Function PointInEllipse( const P : TPoint; const Rect : TRect) : Boolean;
7915:   Function PointInEllipsel( const P: TPoint;Left,Top,Right, Bottom : Integer) : Boolean;
7916:   Function DelphiToLocalFormat( const Format : String) : String
7917:   Function LocalToDelphiFormat( const Format : String) : String
7918:   Procedure TEENableControls(Enable: Boolean; const ControlArray : array of TControl)
7919:   Function TeeRoundDate(const ADate : TDateTime; AStep : TDateTimeStep) : TDateTime
7920:   Procedure TeeDateTimeIncrement(IsDateTime:Boolean;Increment:Boolean;var Value:Double;const
AnIncrement:Double; tmpWhichDateTime:TDateTimeStep)
7921:   Function TeeSortCompare( a, b : Integer) : Integer
7922:   Procedure TeeSortSwap, Procedure ( a, b : Integer)
7923:   Procedure TeeSort(StartIndex,EndIndex:Integer;CompareFunc:TeeSortCompare;SwapFunc:TeeSortSwap);
7924:   Function TeeGetUniqueName( AOwner : TComponent; const AStartName : String) : string
7925:   Function TeeExtractField( St : String; Index : Integer) : String;
7926:   Function TeeExtractField1( St : String; Index : Integer; const Separator : String) : String;
7927:   Function TeeNumFields( St : String) : Integer;
7928:   Function TeeNumFields1( const St, Separator : String) : Integer;
7929:   Procedure TeeGetBitmapEditor( AObject : TObject; var Bitmap : TBitmap)
7930:   Procedure TeeLoadBitmap( Bitmap : TBitmap; const Name1, Name2 : String)
7931:   // TColorArray', 'array of TColor
7932:   Function GetDefaultColor( const Index : Integer) : TColor
7933:   Procedure SetDefaultColorPalette;
7934:   Procedure SetDefaultColorPalettel( const Palette : array of TColor);
7935:   'TeeCheckBoxSize', 'LongInt').SetInt( 11);
7936:   Procedure TeeDrawCheckBox(x,y:Integer;Canvas:TCanvas;Checked:Boolean;ABackColor:TColor;CheckBox:Boolean);
7937:   Function TEEStrToFloatDef( const S : string; const Default : Extended) : Extended
7938:   Function TryStrToFloat( const S : String; var Value : Double) : Boolean

```



```

7939: Function CrossingLines( const X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Double; out x, y : Double) : Boolean
7940: Procedure TeeTranslateControl( AControl : TControl);
7941: Procedure TeeTranslateControl1( AControl : TControl; const ExcludeChilds : array of TControl);
7942: Function ReplaceChar( const AString : String; const Search : Char; const Replace : Char) : String
7943: //Procedure RectToFourPoints( const ARect : TRect; const Angle : Double; var P : TFourPoints)
7944: Function TeeAntiAlias( Panel : TCustomTeePanel; ChartRect : Boolean) : TBitmap
7945: //Procedure DrawBevel(Canvas:TCanvas;Bevel:TPanelBevel;var R:TRect;Width:Integer;Round:Integer);
7946: //Function ScreenRatio( ACanvas : TCanvas3D) : Double
7947: Function TeeReadBoolOption( const AKey : String; DefaultValue : Boolean) : Boolean
7948: Procedure TeeSaveBoolOption( const AKey : String; Value : Boolean)
7949: Function TeeReadIntegerOption( const AKey : String; DefaultValue : Integer) : Integer
7950: Procedure TeeSaveIntegerOption( const AKey : String; Value : Integer)
7951: Function TeeReadStringOption( const AKey, DefaultValue : String) : String
7952: Procedure TeeSaveStringOption( const AKey, Value : String)
7953: Function TeeDefaultXMLEncoding : String
7954: Procedure ConvertTextToXML( Stream : TStream; XMLHeader : Boolean)
7955: TeeWindowHandle', 'Integer
7956: Procedure TeeGotoURL( Handle : TeeWindowHandle; const URL : String)
7957: Procedure HtmlTextOut( ACanvas : TCanvas; x, y : Integer; Text : String)
7958: Function HtmlTextExtent( ACanvas : TCanvas; const Text : String) : TSize
7959: end;
7960:
7961:
7962: using mXBDEUtils
7963: *****
7964: Procedure SetAlias( aAlias, aDirectory : String)
7965: Procedure CheckRegistryEntry(Reg:TRegistry;const Path,Value:String;const Default,
Desired:Variant;Size:Byte);
7966: Function GetFileVersionNumber( const FileName : String) : TVersionNo
7967: Procedure SetBDE( aPath, aNode, aValue : String)
7968: function RestartDialog(Wnd: HWND; Reason: PChar; Flags: Integer): Integer; stdcall;
7969: Function GetSystemDirectory( lpBuffer : string; uSize : UINT) : UINT
7970: Function GetSystemDirectoryW( lpBuffer : pchar; uSize : UINT) : UINT
7971: Function GetTempPath( nBufferLength : DWORD; lpBuffer : string) : DWORD
7972: Function GetWindowsDirectoryW( nBufferLength : DWORD; lpBuffer : string) : DWORD
7973: Function GetTempFileName(lpPathName,lpPrefixString:string;uUnique:UINT;lpTempFileName:string):UINT;
7974:
7975:
7976: procedure SIRegister_cDateTime(CL: TPSPascalCompiler);
7977: begin
7978: AddClassN(FindClass('TOBJECT'),'EDateTime
7979: Function DatePart( const D : TDateTime) : Integer
7980: Function TimePart( const D : TDateTime) : Double
7981: Function Century( const D : TDateTime) : Word
7982: Function Year( const D : TDateTime) : Word
7983: Function Month( const D : TDateTime) : Word
7984: Function Day( const D : TDateTime) : Word
7985: Function Hour( const D : TDateTime) : Word
7986: Function Minute( const D : TDateTime) : Word
7987: Function Second( const D : TDateTime) : Word
7988: Function Millisecond( const D : TDateTime) : Word
7989: ('OneDay','Extended').setExtended( 1.0);
7990: ('OneHour','Extended').SetExtended( OneDay / 24);
7991: ('OneMinute','Extended').SetExtended( OneHour / 60);
7992: ('OneSecond','Extended').SetExtended( OneMinute / 60);
7993: ('OneMillisecond','Extended').SetExtended( OneSecond / 1000);
7994: ('OneWeek','Extended').SetExtended( OneDay * 7);
7995: ('HoursPerDay','Extended').SetExtended( 24);
7996: ('MinutesPerHour','Extended').SetExtended( 60);
7997: ('SecondsPerMinute','Extended').SetExtended( 60);
7998: Procedure SetYear( var D : TDateTime; const Year : Word)
7999: Procedure SetMonth( var D : TDateTime; const Month : Word)
8000: Procedure SetDay( var D : TDateTime; const Day : Word)
8001: Procedure SetHour( var D : TDateTime; const Hour : Word)
8002: Procedure SetMinute( var D : TDateTime; const Minute : Word)
8003: Procedure SetSecond( var D : TDateTime; const Second : Word)
8004: Procedure SetMillisecond( var D : TDateTime; const Milliseconds : Word)
8005: Function IsEqual( const D1, D2 : TDateTime) : Boolean;
8006: Function IsEqual1( const D1 : TDateTime; const Ye, Mo, Da : Word):Boolean;
8007: Function IsEqual2( const D1 : TDateTime; const Ho, Mi, Se, ms : Word):Boolean;
8008: Function IsAM( const D : TDateTime) : Boolean
8009: Function IsPM( const D : TDateTime) : Boolean
8010: Function IsMidnight( const D : TDateTime) : Boolean
8011: Function IsNoon( const D : TDateTime) : Boolean
8012: Function IsSunday( const D : TDateTime) : Boolean
8013: Function IsMonday( const D : TDateTime) : Boolean
8014: Function IsTuesday( const D : TDateTime) : Boolean
8015: Function IsWednesday( const D : TDateTime) : Boolean
8016: Function IsThursday( const D : TDateTime) : Boolean
8017: Function IsFriday( const D : TDateTime) : Boolean
8018: Function IsSaturday( const D : TDateTime) : Boolean
8019: Function IsWeekend( const D : TDateTime) : Boolean
8020: Function Noon( const D : TDateTime) : TDateTime
8021: Function Midnight( const D : TDateTime) : TDateTime
8022: Function FirstDayOfMonth( const D : TDateTime) : TDateTime
8023: Function LastDayOfMonth( const D : TDateTime) : TDateTime
8024: Function NextWorkday( const D : TDateTime) : TDateTime
8025: Function PreviousWorkday( const D : TDateTime) : TDateTime
8026: Function FirstDayOfYear( const D : TDateTime) : TDateTime

```

```

8027: Function LastDayOfYear( const D : TDateTime) : TDateTime
8028: Function EasterSunday( const Year : Word) : TDateTime
8029: Function GoodFriday( const Year : Word) : TDateTime
8030: Function AddMilliseconds( const D : TDateTime; const N : Int64) : TDateTime
8031: Function AddSeconds( const D : TDateTime; const N : Int64) : TDateTime
8032: Function AddMinutes( const D : TDateTime; const N : Integer) : TDateTime
8033: Function AddHours( const D : TDateTime; const N : Integer) : TDateTime
8034: Function AddDays( const D : TDateTime; const N : Integer) : TDateTime
8035: Function AddWeeks( const D : TDateTime; const N : Integer) : TDateTime
8036: Function AddMonths( const D : TDateTime; const N : Integer) : TDateTime
8037: Function AddYears( const D : TDateTime; const N : Integer) : TDateTime
8038: Function DayOfYear( const Ye, Mo, Da : Word) : Integer
8039: Function DayOfYear( const D : TDateTime) : Integer
8040: Function DaysInMonth( const Ye, Mo : Word) : Integer
8041: Function DaysInMonth( const D : TDateTime) : Integer
8042: Function DaysInYear( const Ye : Word) : Integer
8043: Function DaysInYearDate( const D : TDateTime) : Integer
8044: Function WeekNumber( const D : TDateTime) : Integer
8045: Function ISOFirstWeekOfYear( const Ye : Word) : TDateTime
8046: Procedure ISOWeekNumber( const D : TDateTime; var WeekNumber, WeekYear : Word)
8047: Function DiffMilliseconds( const D1, D2 : TDateTime) : Int64
8048: Function DiffSeconds( const D1, D2 : TDateTime) : Integer
8049: Function DiffMinutes( const D1, D2 : TDateTime) : Integer
8050: Function DiffHours( const D1, D2 : TDateTime) : Integer
8051: Function DiffDays( const D1, D2 : TDateTime) : Integer
8052: Function DiffWeeks( const D1, D2 : TDateTime) : Integer
8053: Function DiffMonths( const D1, D2 : TDateTime) : Integer
8054: Function DiffYears( const D1, D2 : TDateTime) : Integer
8055: Function GMTBias : Integer
8056: Function GMTTimeToLocalTime( const D : TDateTime) : TDateTime
8057: Function LocalTimeToGMTTime( const D : TDateTime) : TDateTime
8058: Function NowAsGMTTime : TDateTime
8059: Function DateTimeToISO8601String( const D : TDateTime) : AnsiString
8060: Function ISO8601StringToTime( const D : AnsiString) : TDateTime
8061: Function ISO8601StringAsDateTime( const D : AnsiString) : TDateTime
8062: Function DateTimeToANSI( const D : TDateTime) : Integer
8063: Function ANSIToDateTime( const Julian : Integer) : TDateTime
8064: Function DateTimeToISOInteger( const D : TDateTime) : Integer
8065: Function DateTimeToISOString( const D : TDateTime) : AnsiString
8066: Function ISOIntegerToDateTime( const ISOInteger : Integer) : TDateTime
8067: Function TwoDigitRadix2000YearToYear( const Y : Integer) : Integer
8068: Function DateTimeAsElapsedTime( const D: TDateTime; const IncludeMilliseconds: Boolean) : AnsiString
8069: Function UnixTimeToDateTime( const UnixTime : LongWord) : TDateTime
8070: Function DateTimeToUnixTime( const D : TDateTime) : LongWord
8071: Function EnglishShortDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8072: Function EnglishShortDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8073: Function EnglishLongDayOfWeekStrA( const DayOfWeek : Integer) : AnsiString
8074: Function EnglishLongDayOfWeekStrU( const DayOfWeek : Integer) : UnicodeString
8075: Function EnglishShortMonthStrA( const Month : Integer) : AnsiString
8076: Function EnglishShortMonthStrU( const Month : Integer) : UnicodeString
8077: Function EnglishLongMonthStrA( const Month : Integer) : AnsiString
8078: Function EnglishLongMonthStrU( const Month : Integer) : UnicodeString
8079: Function EnglishShortDayOfWeekA( const S : AnsiString) : Integer
8080: Function EnglishShortDayOfWeekU( const S : UnicodeString) : Integer
8081: Function EnglishLongDayOfWeekA( const S : AnsiString) : Integer
8082: Function EnglishLongDayOfWeekU( const S : UnicodeString) : Integer
8083: Function EnglishShortMonthA( const S : AnsiString) : Integer
8084: Function EnglishShortMonthU( const S : UnicodeString) : Integer
8085: Function EnglishLongMonthA( const S : AnsiString) : Integer
8086: Function EnglishLongMonthU( const S : UnicodeString) : Integer
8087: Function RFC850DayOfWeekA( const S : AnsiString) : Integer
8088: Function RFC850DayOfWeekU( const S : UnicodeString) : Integer
8089: Function RFC1123DayOfWeekA( const S : AnsiString) : Integer
8090: Function RFC1123DayOfWeekU( const S : UnicodeString) : Integer
8091: Function RFCMonthA( const S : AnsiString) : Word
8092: Function RFCMonthU( const S : UnicodeString) : Word
8093: Function GMTTimeToRFC1123TimeA( const D : TDateTime; const IncludeSeconds: Boolean) : AnsiString
8094: Function GMTTimeToRFC1123TimeU( const D : TDateTime; const IncludeSeconds: Boolean) : UnicodeString
8095: Function GMTDateTimeToRFC1123DateTimeA( const D: TDateTime; const IncludeDayOfWeek: Bool) : AnsiString;
8096: Function GMTDateTimeToRFC1123DateTimeU( const D: TDateTime; const IncludeDayOfWeek: Bool) : UnicodeString;
8097: Function DateTimeToRFCDateTimeA( const D : TDateTime) : AnsiString
8098: Function DateTimeToRFCDateTimeU( const D : TDateTime) : UnicodeString
8099: Function NowAsRFCDateTimeA : AnsiString
8100: Function NowAsRFCDateTimeU : UnicodeString
8101: Function RFCDateTimeToGMTDateTime( const S : AnsiString) : TDateTime
8102: Function RFCDateTimeToDateTime( const S : AnsiString) : TDateTime
8103: Function RFCTimeZoneToGMTBias( const Zone : AnsiString) : Integer
8104: Function TimePeriodStr( const D : TDateTime) : AnsiString
8105: Procedure SelfTest
8106: end;
8107: //*****CFFileUtils
8108: Function PathHasDriveLetterA( const Path : AnsiString) : Boolean
8109: Function PathHasDriveLetter( const Path : String) : Boolean
8110: Function PathIsDriveLetterA( const Path : AnsiString) : Boolean
8111: Function PathIsDriveLetter( const Path : String) : Boolean
8112: Function PathIsDriveRootA( const Path : AnsiString) : Boolean
8113: Function PathIsDriveRoot( const Path : String) : Boolean
8114: Function PathIsRootA( const Path : AnsiString) : Boolean
8115: Function PathIsRoot( const Path : String) : Boolean

```

```

8116: Function PathIsUNCPathA( const Path : AnsiString) : Boolean
8117: Function PathIsUNCPath( const Path : String) : Boolean
8118: Function PathIsAbsoluteA( const Path : AnsiString) : Boolean
8119: Function PathIsAbsolute( const Path : String) : Boolean
8120: Function PathIsDirectoryA( const Path : AnsiString) : Boolean
8121: Function PathIsDirectory( const Path : String) : Boolean
8122: Function PathInclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString
8123: Function PathInclSuffix( const Path : String; const PathSep : Char) : String
8124: Function PathExclSuffixA( const Path : AnsiString; const PathSep : Char) : AnsiString
8125: Function PathExclSuffix( const Path : String; const PathSep : Char) : String
8126: Procedure PathEnsureSuffixA( var Path : AnsiString; const PathSep : Char)
8127: Procedure PathEnsureSuffix( var Path : String; const PathSep : Char)
8128: Procedure PathEnsureNoSuffixA( var Path : AnsiString; const PathSep : Char)
8129: Procedure PathEnsureNoSuffix( var Path : String; const PathSep : Char)
8130: //Function PathCanonicalA( const Path : AnsiString; const PathSep : Char) : AnsiString
8131: Function PathCanonical( const Path : String; const PathSep : Char) : String
8132: Function PathExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8133: Function PathExpand( const Path : String; const BasePath : String; const PathSep : Char) : String
8134: Function PathLeftElementA( const Path : AnsiString; const PathSep : Char) : AnsiString
8135: Function PathLeftElement( const Path : String; const PathSep : Char) : String
8136: Procedure PathSplitLeftElementA(const Path:AString;var LeftElement,RightPath:AString;const PathSep:Char);
8137: Procedure PathSplitLeftElement(const Path:String; var LeftElement,RightPath: String;const PathSep:Char);
8138: Procedure DecodeFilePathA(const FilePath:AnsiString; var Path,FileName:AnsiString;const PathSep:Char;
8139: Procedure DecodeFilePath( const FilePath : String; var Path, FileName : String; const PathSep : Char)
8140: Function FileNameValidA( const FileName : AnsiString) : AnsiString
8141: Function FileNameValid( const FileName : String) : String
8142: Function FilePathA(const FileName,Path:AnsiString;const BasePath:AnsiStr;const PathSep:Char):AnsiString;
8143: Function FilePath(const FileName, Path: String;const BasePath: String;const PathSep : Char) : String
8144: Function DirectoryExpandA(const Path:AnsiString;const BasePath:AnsiString;const PathSep:Char):AnsiString
8145: Function DirectoryExpand(const Path: String; const BasePath: String; const PathSep : Char) : String
8146: Function UnixPathToWinPath( const Path : AnsiString) : AnsiString
8147: Function WinPathToUnixPath( const Path : AnsiString) : AnsiString
8148: Procedure CCopyFile( const FileName, DestName : String)
8149: Procedure CMoveFile( const FileName, DestName : String)
8150: Function CDeleteFiles( const FileMask : String) : Boolean
8151: Function FileSeekEx(const FHandle:TFileHandle;const FileOffset:Int64; const FilePos:TFileSeekPos):Int64;
8152: Procedure FileCloseEx( const FileHandle : TFileHandle)
8153: Function FileExistsA( const FileName : AnsiString) : Boolean
8154: Function FileExists( const FileName : String) : Boolean
8155: Function CFileGetSize( const FileName : String) : Int64
8156: Function FileGetDateTime( const FileName : String) : TDateTime
8157: Function FileGetDateTime2( const FileName : String) : TDateTime
8158: Function FileIsReadOnly( const FileName : String) : Boolean
8159: Procedure FileDeleteEx( const FileName : String)
8160: Procedure FileRenameEx( const OldFileName, NewFileName : String)
8161: Function ReadFileStrA( const FileName:AnsiString; const FileSharing : TFileSharing; const FileCreationMode
: TFileCreationMode; const FileOpenWait : PFileOpenWait) : AnsiString
8162: Function DirectoryEntryExists( const Name : String) : Boolean
8163: Function DirectoryEntrySize( const Name : String) : Int64
8164: Function CDirectoryExists( const DirectoryName : String) : Boolean
8165: Function DirectoryGetDateTime( const DirectoryName : String) : TDateTime
8166: Procedure CDirectoryCreate( const DirectoryName : String)
8167: Function GetFirstFileNameMatching( const FileMask : String) : String
8168: Function DirEntryGetAttr( const FileName : AnsiString) : Integer
8169: Function DirEntryIsDirectory( const FileName : AnsiString) : Boolean
8170: Function FileHasAttr( const FileName : String; const Attr : Word) : Boolean
8171: AddTypeS('TLogicalDriveType', '( DriveRemovable, DriveFixed, DriveRemote, '
8172: + 'DriveCDRom, DriveRamDisk, DriveTypeUnknown )
8173: Function DriveIsValid( const Drive : Char) : Boolean
8174: Function DriveGetType( const Path : AnsiString) : TLogicalDriveType
8175: Function DriveFreeSpace( const Path : AnsiString) : Int64
8176:
8177: procedure SIRegister_cTimers(CL: TPSPascalCompiler);
8178: begin
8179: AddClassN(FindClass('TOBJECT'),'ETimers
8180: Const('TickFrequency','LongInt').SetInt( 1000);Function GetTick : LongWord
8181: Function TickDelta( const D1, D2 : LongWord) : Integer
8182: Function TickDeltaW( const D1, D2 : LongWord) : LongWord
8183: AddTypeS('THPTimer', 'Int64
8184: Procedure StartTimer( var Timer : THPTimer)
8185: Procedure StopTimer( var Timer : THPTimer)
8186: Procedure ResumeTimer( var StoppedTimer : THPTimer)
8187: Procedure InitStoppedTimer( var Timer : THPTimer)
8188: Procedure InitElapsedTimer( var Timer : THPTimer; const Milliseconds : Integer)
8189: Function MillisecondsElapsed( const Timer : THPTimer; const TimerRunning : Boolean) : Integer
8190: Function MicrosecondsElapsed( const Timer : THPTimer; const TimerRunning : Boolean) : Int64
8191: Procedure WaitMicroseconds( const MicroSeconds : Integer)
8192: Function GetHighPrecisionFrequency : Int64
8193: Function GetHighPrecisionTimerOverhead : Int64
8194: Procedure AdjustTimerForOverhead( var StoppedTimer : THPTimer; const Overhead : Int64)
8195: Procedure SelfTestCTimer
8196: end;
8197:
8198: procedure SIRegister_cRandom(CL: TPSPascalCompiler);
8199: begin
8200: Function RandomSeed : LongWord
8201: Procedure AddEntropy( const Value : LongWord)
8202: Function RandomUniform : LongWord;
8203: Function RandomUniform1( const N : Integer) : Integer;

```

```

8204: Function RandomBoolean : Boolean
8205: Function RandomByte : Byte
8206: Function RandomByteNonZero : Byte
8207: Function RandomWord : Word
8208: Function RandomInt64 : Int64;
8209: Function RandomInt64I( const N : Int64) : Int64;
8210: Function RandomHex( const Digits : Integer) : String
8211: Function RandomFloat : Extended
8212: Function RandomAlphaStr( const Length : Integer) : AnsiString
8213: Function RandomPseudoWord( const Length : Integer) : AnsiString
8214: Function RandomPassword(const MinL,MaxLength:Int;const CaseSens,UseSymbols,UseNumbers:Bool):AnsiString;
8215: Function mwcRandomLongWord : LongWord
8216: Function urnRandomLongWord : LongWord
8217: Function moaRandomFloat : Extended
8218: Function mwcRandomFloat : Extended
8219: Function RandomNormalF : Extended
8220: Procedure SelfTestCRandom
8221: end;
8222:
8223: procedure SIRegister_SynEditMiscProcs(CL: TPSPascalCompiler);
8224: begin
8225:   // PIntArray, 'TIntArray // will not work
8226:   Addtypes('TConvertTabsProc','function(const Line:AnsiString; TabWidth: integer):AnsiString
8227:   Addtypes('TConvertTabsProcEx','function(const Line:AnsiString; TabWidth: integer;var HasTabs: boolean):
AnsiString
8228:   Function synMax( x, y : integer) : integer
8229:   Function synMin( x, y : integer) : integer
8230:   Function synMinMax( x, mi, ma : integer) : integer
8231:   Procedure synSwapInt( var l, r : integer)
8232:   Function synMaxPoint( const P1, P2 : TPoint) : TPoint
8233:   Function synMinPoint( const P1, P2 : TPoint) : TPoint
8234:   //Function synGetIntArray( Count : Cardinal; InitialValue : integer) : PIntArray
8235:   Procedure synInternalFillRect( dc : HDC; const rcPaint : TRect)
8236:   Function synGetBestConvertTabsProc( TabWidth : integer) : TConvertTabsProc
8237:   Function synConvertTabs( const Line : AnsiString; TabWidth : integer) : AnsiString
8238:   Function synGetBestConvertTabsProcEx( TabWidth : integer) : TConvertTabsProcEx
8239:   Function synConvertTabsEx(const Line:AnsiString;TabWidth:integer; var HasTabs:boolean):AnsiString;
8240:   Function synGetExpandedLength( const aStr : string; aTabWidth : integer) : integer
8241:   Function synCharIndex2CaretPos( Index, TabWidth : integer; const Line : string) : integer
8242:   Function synCaretPos2CharIndex(Position,TabWidth:int;const Line:string;var InsideTabChar:boolean):int;
8243:   Function synStrScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8244:   Function synStrRScanForCharInSet(const Line:string;Start:integer;AChars:TSynIdentChars):integer;
8245:   TStringType', '( stNone, stHalfNumAlpha, stHalfSymbol, stHalfKat'
8246:   + 'akana,stWideNumAlpha,stWideSymbol,stWideKatakana,stHiragana,stIdeograph,stControl,stKashida )
8247:   ('C3_NONSPACING','LongInt').SetInt( 1);
8248:   'C3_DIACRITIC','LongInt').SetInt( 2);
8249:   'C3_VOWELMARK','LongInt').SetInt( 4);
8250:   ('C3_SYMBOL','LongInt').SetInt( 8);
8251:   ('C3_KATAKANA','LongWord').SetUInt( $0010);
8252:   ('C3_HIRAGANA','LongWord').SetUInt( $0020);
8253:   ('C3_HALFWIDTH','LongWord').SetUInt( $0040);
8254:   ('C3_FULLWIDTH','LongWord').SetUInt( $0080);
8255:   ('C3_IDEOGRAPH','LongWord').SetUInt( $0100);
8256:   ('C3_KASHIDA','LongWord').SetUInt( $0200);
8257:   ('C3_LEXICAL','LongWord').SetUInt( $0400);
8258:   ('C3_ALPHA','LongWord').SetUInt( $8000);
8259:   ('C3_NOTAPPLICABLE','LongInt').SetInt( 0);
8260:   Function synStrScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8261:   Function synStrRScanForMultiByteChar( const Line : string; Start : Integer) : Integer
8262:   Function synIsStringType( Value : Word) : TStringType
8263:   Function synGetEOL( Line : PChar) : PChar
8264:   Function synEncodeString( s : string) : string
8265:   Function synDecodeString( s : string) : string
8266:   Procedure synFreeAndNil( var Obj: TObject)
8267:   Procedure synAssert( Expr : Boolean)
8268:   Function synLastDelimiter( const Delimiters, S : string) : Integer
8269:   TReplaceFlag', '( rfReplaceAll, rfIgnoreCase )
8270:   TReplaceFlags', 'set of TReplaceFlag )
8271:   Function synStringReplace(const S, OldPattern, NewPattern : string; Flags: TReplaceFlags) : string
8272:   Function synGetRValue( RGBValue : TColor) : byte
8273:   Function synGetGValue( RGBValue : TColor) : byte
8274:   Function synGetBValue( RGBValue : TColor) : byte
8275:   Function synRGB( r, g, b : Byte) : Cardinal
8276:   // THighlighterAttriProc', 'Function ( Highlighter : TSynCustomHigh'
8277:   // + 'lighter; Attr:TSynHighlighterAttributes;UniqueAttriName:string;Params array of Pointer):Boolean;
8278:   //Function synEnumHighlighterAttris( Highlighter : TSynCustomHighlighter; SkipDuplicates : Boolean;
HighlighterAttriProc : THighlighterAttriProc; Params : array of Pointer) : Boolean
8279:   Function synCalcPCS( const ABuf, ABufSize : Cardinal) : Word
8280:   Procedure synSynDrawGradient(const ACanvas:TCanvas;const AStartColor,
AEndColor:TColor;ASteps:integer;const ARect : TRect; const AHorizontal : boolean)
8281: end;
8282:
8283: Function GET_APPCOMMAND_LPARAM( lParam : LPARAM) : WORD
8284: Function GET_DEVICE_LPARAM( lParam : LPARAM) : WORD
8285: Function GET_KEYSTATE_LPARAM( lParam : LPARAM) : WORD
8286:
8287: procedure SIRegister_synutil(CL: TPSPascalCompiler);
8288: begin
8289:   Function STimeZoneBias : integer

```



```

8290: Function TimeZone : string
8291: Function Rfc822DateTime( t : TDateTime) : string
8292: Function CDateTime( t : TDateTime) : string
8293: Function SimpleDateTime( t : TDateTime) : string
8294: Function AnsiCDateTime( t : TDateTime) : string
8295: Function GetMonthNumber( Value : string) : integer
8296: Function GetTimeFromStr( Value : string) : TDateTime
8297: Function GetDateMDYFromStr( Value : string) : TDateTime
8298: Function DecodeRfcDateTime( Value : string) : TDateTime
8299: Function GetUTTime : TDateTime
8300: Function SetUTTime( Newdt : TDateTime) : Boolean
8301: Function SGetTick : LongWord
8302: Function SStickDelta( TickOld, TickNew : LongWord) : LongWord
8303: Function CodeInt( Value : Word) : AnsiString
8304: Function DecodeInt( const Value : AnsiString; Index : Integer) : Word
8305: Function CodeLongInt( Value : LongInt) : AnsiString
8306: Function DecodeLongInt( const Value : AnsiString; Index : Integer) : LongInt
8307: Function DumpStr( const Buffer : AnsiString) : string
8308: Function DumpExStr( const Buffer : AnsiString) : string
8309: Procedure Dump( const Buffer : AnsiString; DumpFile : string)
8310: Procedure DumpEx( const Buffer : AnsiString; DumpFile : string)
8311: Function TrimSPLeft( const S : string) : string
8312: Function TrimSPRight( const S : string) : string
8313: Function TrimSP( const S : string) : string
8314: Function SeparateLeft( const Value, Delimiter : string) : string
8315: Function SeparateRight( const Value, Delimiter : string) : string
8316: Function SGetParameter( const Value, Parameter : string) : string
8317: Procedure ParseParametersEx( Value, Delimiter : string; const Parameters : TStrings)
8318: Procedure ParseParameters( Value : string; const Parameters : TStrings)
8319: Function IndexByBegin( Value : string; const List : TStrings) : integer
8320: Function GetEmailAddr( const Value : string) : string
8321: Function GetEmailDesc( Value : string) : string
8322: Function CStrToHex( const Value : AnsiString) : string
8323: Function CIntToBin( Value : Integer; Digits : Byte) : string
8324: Function CBinToInt( const Value : string) : Integer
8325: Function ParseURL( URL : string; var Prot, User, Pass, Host, Port, Path, Para: string): string
8326: Function CReplaceString( Value, Search, Replace : AnsiString) : AnsiString
8327: Function CRPosEx( const Sub, Value : string; From : integer) : Integer
8328: Function CRPos( const Sub, Value : String) : Integer
8329: Function FetchBin( var Value : string; const Delimiter : string) : string
8330: Function CFetch( var Value : string; const Delimiter : string) : string
8331: Function FetchEx( var Value : string; const Delimiter, Quotation : string) : string
8332: Function IsBinaryString( const Value : AnsiString) : Boolean
8333: Function PosCRLF( const Value : AnsiString; var Terminator : AnsiString) : integer
8334: Procedure StringsTrim( const value : TStrings)
8335: Function PosFrom( const SubStr, Value : String; From : integer) : integer
8336: Function IncPoint( const p : __pointer; Value : integer) : __pointer
8337: Function GetBetween( const PairBegin, PairEnd, Value : string) : string
8338: Function CCountOfChar( const Value : string; aChr : char) : integer
8339: Function UnquoteStr( const Value : string; Quote : Char) : string
8340: Function QuoteStr( const Value : string; Quote : Char) : string
8341: Procedure HeadersToList( const Value : TStrings)
8342: Procedure ListToHeaders( const Value : TStrings)
8343: Function SwapBytes( Value : integer) : integer
8344: Function ReadStrFromStream( const Stream : TStream; len : integer) : AnsiString
8345: Procedure WriteStrToStream( const Stream : TStream; Value : AnsiString)
8346: Function GetTempFile( const Dir, prefix : AnsiString) : AnsiString
8347: Function CPadString( const Value : AnsiString; len : integer; Pad : AnsiChar) : AnsiString
8348: Function CXorString( Indata1, Indata2 : AnsiString) : AnsiString
8349: Function NormalizeHeader( Value : TStrings; var Index : Integer) : string
8350: end;
8351:
8352: procedure SIRegister_StCRC(CL: TPSPascalCompiler);
8353: begin
8354:   ('CrcBufSize', 'LongInt').SetInt( 2048);
8355:   Function Adler32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8356:   Function Adler32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8357:   Function Adler32OfFile( FileName : AnsiString) : LongInt
8358:   Function Crc16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8359:   Function Crc16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8360:   Function Crc16OfFile( FileName : AnsiString) : Cardinal
8361:   Function Crc32Prim( var Data, DataSize : Cardinal; CurCrc : LongInt) : LongInt
8362:   Function Crc32OfStream( Stream : TStream; CurCrc : LongInt) : LongInt
8363:   Function Crc32OfFile( FileName : AnsiString) : LongInt
8364:   Function InternetSumPrim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8365:   Function InternetSumOfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8366:   Function InternetSumOfFile( FileName : AnsiString) : Cardinal
8367:   Function Kermit16Prim( var Data, DataSize, CurCrc : Cardinal) : Cardinal
8368:   Function Kermit16OfStream( Stream : TStream; CurCrc : Cardinal) : Cardinal
8369:   Function Kermit16OfFile( FileName : AnsiString) : Cardinal
8370: end;
8371:
8372: procedure SIRegister_ComObj(c1: TPSPascalCompiler);
8373: begin
8374:   function CreateOleObject(const ClassName : String) : IDispatch;
8375:   function GetActiveOleObject(const ClassName : String) : IDispatch;
8376:   function ProgIDToClassID(const ProgID : string) : TGUID;
8377:   function ClassIDToProgID(const ClassID : TGUID) : string;
8378:   function CreateClassID : string;

```

```

8379: function CreateGUIDString: string;
8380: function CreateGUIDID: string;
8381: procedure OleError(ErrorCode: longint);
8382: procedure OleCheck(Result: HRESULT);
8383: end;
8384:
8385: Function xCreateOleObject( const ClassName : string) : Variant //or IDispatch
8386: Function xGetActiveOleObject( const ClassName : string) : Variant
8387: //Function DllGetClassObject( const CLSID : TCLSID; const IID : TIID; var Obj) : HRESULT
8388: Function DllCanUnloadNow : HRESULT
8389: Function DllRegisterServer : HRESULT
8390: Function DllUnregisterServer : HRESULT
8391: Function VarFromInterface( Unknown : IUnknown) : Variant
8392: Function VarToInterface( const V : Variant) : IDispatch
8393: Function VarToAutoObject( const V : Variant) : TAutoObject
8394: //Procedure
DispInvoke(Dispatch:IDispatch;CallDesc:PCallDesc;DispIDs:PDispIDList;Params:Pointer;Res:PVariant);
8395: //Procedure DispInvokeError( Status : HRESULT; const ExcepInfo : TExcepInfo)
8396: Procedure OleError( ErrorCode : HRESULT)
8397: Procedure OleCheck( Result : HRESULT)
8398: Function StringToClassID( const S : string) : TCLSID
8399: Function ClassIDToString( const ClassID : TCLSID) : string
8400: Function xProgIDToClassID( const ProgID : string) : TCLSID
8401: Function xClassIDToProgID( const ClassID : TCLSID) : string
8402: Function xWideCompareStr( const S1, S2 : WideString) : Integer
8403: Function xWideSameStr( const S1, S2 : WideString) : Boolean
8404: Function xGUIDToString( const ClassID : TGUID) : string
8405: Function xStringToGUID( const S : string) : TGUID
8406: Function xGetModuleName( Module : HMODULE) : string
8407: Function xAcquireExceptionObject : TObject
8408: Function xIfThen( AValue : Boolean; const ATrue : Integer; const AFalse : Integer) : Integer
8409: Function xUtf8Encode( const WS : WideString) : UTF8String
8410: Function xUtf8Decode( const S : UTF8String) : WideString
8411: Function xExcludeTrailingPathDelimiter( const S : string) : string
8412: Function xIncludeTrailingPathDelimiter( const S : string) : string
8413: Function XRTLHandleCOMException : HRESULT
8414: Procedure XRTLCheckArgument( Flag : Boolean)
8415: //Procedure XRTLCheckOutArgument( out Arg)
8416: Procedure XRTLInterfaceConnect(const Source:IUnknown; const IID:TIID;const Sink:IUnknown;var
Connection:Longint);
8417: Procedure XRTLInterfaceDisconnect(const Source: IUnknown; const IID:TIID;var Connection : Longint)
8418: Function XRTLRegisterActiveObject(const Unk:IUnknown;ClassID:TCLSID;Flags:DWORD;var
RegisterCookie:Int):HRESULT
8419: Function XRTLUnRegisterActiveObject( var RegisterCookie : Integer) : HRESULT
8420: //Function XRTLGetActiveObject( ClassID : TCLSID; RIID : TIID; out Obj) : HRESULT
8421: Procedure XRTLEnumActiveObjects( Strings : TStrings)
8422: function XRTLDefaultCategoryManager: IUnknown;
8423: function XRTLIsCategoryEmpty(CatID: TGUID; const CategoryManager: IUnknown = nil): Boolean;
8424: // ICatRegister helper functions
8425: function XRTLCreateComponentCategory(CatID: TGUID; CatDescription: WideString;
8426: LocaleID: TLCID = LOCALE_USER_DEFAULT;
8427: const CategoryManager: IUnknown = nil): HRESULT;
8428: function XRTLRemoveComponentCategory(CatID: TGUID; CatDescription: WideString;
8429: LocaleID: TLCID = LOCALE_USER_DEFAULT;
8430: const CategoryManager: IUnknown = nil): HRESULT;
8431: function XRTLRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8432: const CategoryManager: IUnknown = nil): HRESULT;
8433: function XRTLUnRegisterCLSIDInCategory(ClassID: TGUID; CatID: TGUID;
8434: const CategoryManager: IUnknown = nil): HRESULT;
8435: // ICatInformation helper functions
8436: function XRTLGetCategoryDescription(CatID: TGUID; var CatDescription: WideString;
8437: LocaleID: TLCID = LOCALE_USER_DEFAULT;
8438: const CategoryManager: IUnknown = nil): HRESULT;
8439: function XRTLGetCategoryList(Strings: TStrings; LocaleID: TLCID = LOCALE_USER_DEFAULT;
8440: const CategoryManager: IUnknown = nil): HRESULT;
8441: function XRTLGetCategoryCLSIDList(CatID: TGUID; Strings: TStrings;
8442: const CategoryManager: IUnknown = nil): HRESULT;
8443: function XRTLGetCategoryProgIDList(CatID: TGUID; Strings: TStrings;
8444: const CategoryManager: IUnknown = nil): HRESULT;
8445: function XRTLFetch(var AInput: WideString; const ADelim: WideString = ' ';
8446: const ADelete: Boolean = True): WideString;
8447: function XRTLRePos(const ASub, AIn: WideString; AStart: Integer = -1): Integer;
8448: function XRTLGetVariantAsString( const Value : Variant) : string
8449: Function XRTLDateTimeToTimeZoneTime( DT : TDateTime; TimeZone : TXRTLTimeZone) : TDateTime
8450: Function XRTLGetTimeZones : TXRTLTimeZones
8451: Function XFileTimeToDateTime( FileTime : TFileTime) : TDateTime
8452: Function DateTimeToFileTime( DateTime : TDateTime) : TFileTime
8453: Function GMTNow : TDateTime
8454: Function GMTToLocalTime( GMT : TDateTime) : TDateTime
8455: Function LocalTimeToGMT( LocalTime : TDateTime) : TDateTime
8456: Procedure XRTLNotImplemented
8457: Procedure XRTLRaiseError( E : Exception)
8458: Procedure XRTLInvalidOperation( ClassName:string; OperationName:string; Description: string)
8459:
8460:
8461: procedure SIRegister_xrtl_util_Value(CL: TPSPascalCompiler);
8462: begin
8463: SIRegister_IXRTLValue(CL);
8464: SIRegister_TXRTLValue(CL);

```

```

8465: //AddTypeS('PXRTLValueArray', '^TXRTLValueArray // will not work
8466: AddTypeS('TXRTLValueArray', 'array of IXRTLValue
8467: Function XRTLValue( const AValue : Cardinal) : IXRTLValue;
8468: Function XRTLSetValue( const IValue : IXRTLValue; const AValue : Cardinal) : Cardinal;
8469: Function XRTLGetAsCardinal( const IValue : IXRTLValue) : Cardinal
8470: Function XRTLGetAsCardinalDef( const IValue : IXRTLValue; const DefValue : Cardinal) : Cardinal
8471: Function XRTLValue1( const AValue : Integer) : IXRTLValue;
8472: Function XRTLSetValue1( const IValue : IXRTLValue; const AValue : Integer) : Integer;
8473: Function XRTLGetAsInteger( const IValue : IXRTLValue) : Integer
8474: Function XRTLGetAsIntegerDef( const IValue : IXRTLValue; const DefValue : Integer) : Integer
8475: Function XRTLValue2( const AValue : Int64) : IXRTLValue;
8476: Function XRTLSetValue2( const IValue : IXRTLValue; const AValue : Int64) : Int64;
8477: Function XRTLGetAsInt64( const IValue : IXRTLValue) : Int64
8478: Function XRTLGetAsInt64Def( const IValue : IXRTLValue; const DefValue : Int64) : Int64
8479: Function XRTLValue3( const AValue : Single) : IXRTLValue;
8480: Function XRTLSetValue3( const IValue : IXRTLValue; const AValue : Single) : Single;
8481: Function XRTLGetAsSingle( const IValue : IXRTLValue) : Single
8482: Function XRTLGetAsSingleDef( const IValue : IXRTLValue; const DefValue : Single) : Single
8483: Function XRTLValue4( const AValue : Double) : IXRTLValue;
8484: Function XRTLSetValue4( const IValue : IXRTLValue; const AValue : Double) : Double;
8485: Function XRTLGetAsDouble( const IValue : IXRTLValue) : Double
8486: Function XRTLGetAsDoubleDef( const IValue : IXRTLValue; const DefValue : Double) : Double
8487: Function XRTLValue5( const AValue : Extended) : IXRTLValue;
8488: Function XRTLSetValue5( const IValue : IXRTLValue; const AValue : Extended) : Extended;
8489: Function XRTLGetAsExtended( const IValue : IXRTLValue) : Extended
8490: Function XRTLGetAsExtendedDef( const IValue : IXRTLValue; const DefValue : Extended) : Extended
8491: Function XRTLValue6( const AValue : IInterface) : IXRTLValue;
8492: Function XRTLSetValue6( const IValue : IXRTLValue; const AValue : IInterface) : IInterface;
8493: Function XRTLGetAsInterface( const IValue : IXRTLValue) : IInterface;
8494: //Function XRTLGetAsInterface1( const IValue : IXRTLValue; out Obj : IInterface;
8495: Function XRTLGetAsInterfaceDef( const IValue : IXRTLValue; const DefValue : IInterface) : IInterface;
8496: Function XRTLValue7( const AValue : WideString) : IXRTLValue;
8497: Function XRTLSetValue7( const IValue : IXRTLValue; const AValue : WideString) : WideString;
8498: Function XRTLGetAsWideString( const IValue : IXRTLValue) : WideString
8499: Function XRTLGetAsWideStringDef( const IValue : IXRTLValue; const DefValue : WideString) : WideString
8500: Function XRTLValue8( const AValue : TObject; const AOwnValue : Boolean) : IXRTLValue;
8501: Function XRTLSetValue8( const IValue : IXRTLValue; const AValue : TObject) : TObject;
8502: Function XRTLGetAsObject( const IValue : IXRTLValue; const ADetachOwnership : Boolean) : TObject;
8503: Function XRTLGetAsObjectDef( const IValue : IXRTLValue; const DefValue : TObject; const
ADetachOwnership : Boolean) : TObject;
8504: //Function XRTLValue9( const AValue : __Pointer) : IXRTLValue;
8505: //Function XRTLSetValue9( const IValue : IXRTLValue; const AValue : __Pointer) : __Pointer;
8506: //Function XRTLGetAsPointer( const IValue : IXRTLValue) : __Pointer
8507: //Function XRTLGetAsPointerDef( const IValue : IXRTLValue; const DefValue : __Pointer) : __Pointer
8508: Function XRTLValueV( const AValue : Variant) : IXRTLValue;
8509: Function XRTLSetValueV( const IValue : IXRTLValue; const AValue : Variant) : Variant;
8510: Function XRTLGetAsVariant( const IValue : IXRTLValue) : Variant
8511: Function XRTLGetAsVariantDef( const IValue : IXRTLValue; const DefValue : Variant) : Variant
8512: Function XRTLValue10( const AValue : Currency) : IXRTLValue;
8513: Function XRTLSetValue10( const IValue : IXRTLValue; const AValue : Currency) : Currency;
8514: Function XRTLGetAsCurrency( const IValue : IXRTLValue) : Currency
8515: Function XRTLGetAsCurrencyDef( const IValue : IXRTLValue; const DefValue : Currency) : Currency
8516: Function XRTLValue11( const AValue : Comp) : IXRTLValue;
8517: Function XRTLSetValue11( const IValue : IXRTLValue; const AValue : Comp) : Comp;
8518: Function XRTLGetAsComp( const IValue : IXRTLValue) : Comp
8519: Function XRTLGetAsCompDef( const IValue : IXRTLValue; const DefValue : Comp) : Comp
8520: Function XRTLValue12( const AValue : TClass) : IXRTLValue;
8521: Function XRTLSetValue12( const IValue : IXRTLValue; const AValue : TClass) : TClass;
8522: Function XRTLGetAsClass( const IValue : IXRTLValue) : TClass
8523: Function XRTLGetAsClassDef( const IValue : IXRTLValue; const DefValue : TClass) : TClass
8524: Function XRTLValue13( const AValue : TGUID) : IXRTLValue;
8525: Function XRTLSetValue13( const IValue : IXRTLValue; const AValue : TGUID) : TGUID;
8526: Function XRTLGetAsGUID( const IValue : IXRTLValue) : TGUID
8527: Function XRTLGetAsGUIDDef( const IValue : IXRTLValue; const DefValue : TGUID) : TGUID
8528: Function XRTLValue14( const AValue : Boolean) : IXRTLValue;
8529: Function XRTLSetValue14( const IValue : IXRTLValue; const AValue : Boolean) : Boolean;
8530: Function XRTLGetAsBoolean( const IValue : IXRTLValue) : Boolean
8531: Function XRTLGetAsBooleanDef( const IValue : IXRTLValue; const DefValue : Boolean) : Boolean
8532: end;
8533:
8534: //*****unit uPSI_GR32;*****
8535:
8536: Function Color32( WinColor : TColor) : TColor32;
8537: Function Color321( R, G, B : Byte; A : Byte) : TColor32;
8538: Function Color322( Index : Byte; var Palette : TPalette32) : TColor32;
8539: Function Gray32( Intensity : Byte; Alpha : Byte) : TColor32
8540: Function WinColor( Color32 : TColor32) : TColor
8541: Function ArrayOfColor32( Colors : array of TColor32) : TArrayOfColor32
8542: Procedure Color32ToRGB( Color32 : TColor32; var R, G, B : Byte)
8543: Procedure Color32ToRGBA( Color32 : TColor32; var R, G, B, A : Byte)
8544: Function Color32Components( R, G, B, A : Boolean) : TColor32Components
8545: Function RedComponent( Color32 : TColor32) : Integer
8546: Function GreenComponent( Color32 : TColor32) : Integer
8547: Function BlueComponent( Color32 : TColor32) : Integer
8548: Function AlphaComponent( Color32 : TColor32) : Integer
8549: Function Intensity( Color32 : TColor32) : Integer
8550: Function SetAlpha( Color32 : TColor32; NewAlpha : Integer) : TColor32
8551: Function HSLtoRGB( H, S, L : Single) : TColor32;
8552: Procedure RGBtoHSL( RGB : TColor32; out H, S, L : Single);

```

```

8553: Function HSLtoRGB1( H, S, L : Integer ) : TColor32;
8554: Procedure RGBtoHSL1( RGB : TColor32; out H, S, L : Byte);
8555: Function WinPalette( const P : TPalette32 ) : HPALETTE
8556: Function FloatPoint( X, Y : Single ) : TFloatPoint;
8557: Function FloatPoint1( const P : TPoint ) : TFloatPoint;
8558: Function FloatPoint2( const FXP : TFixedPoint ) : TFloatPoint;
8559: Function FixedPoint( X, Y : Integer ) : TFixedPoint;
8560: Function FixedPoint1( X, Y : Single ) : TFixedPoint;
8561: Function FixedPoint2( const P : TPoint ) : TFixedPoint;
8562: Function FixedPoint3( const FP : TFloatPoint ) : TFixedPoint;
8563: AddTypes( 'TRectRounding', '( rrClosest, rrOutside, rrInside )
8564: Function MakeRect( const L, T, R, B : Integer ) : TRect;
8565: Function MakeRect1( const FR : TFloatRect; Rounding : TRectRounding ) : TRect;
8566: Function MakeRect2( const FXR : TRect; Rounding : TRectRounding ) : TRect;
8567: Function GFixedRect( const L, T, R, B : TFixed ) : TRect;
8568: Function FixedRect1( const ARect : TRect ) : TRect;
8569: Function FixedRect2( const FR : TFloatRect ) : TRect;
8570: Function GFloatRect( const L, T, R, B : TFloat ) : TFloatRect;
8571: Function FloatRect1( const ARect : TRect ) : TFloatRect;
8572: Function FloatRect2( const FXR : TRect ) : TFloatRect;
8573: Function GIntersectRect( out Dst : TRect; const R1, R2 : TRect ) : Boolean;
8574: Function IntersectRect1( out Dst : TFloatRect; const FR1, FR2 : TFloatRect ) : Boolean;
8575: Function GUnionRect( out Rect : TRect; const R1, R2 : TRect ) : Boolean;
8576: Function UnionRect1( out Rect : TFloatRect; const R1, R2 : TFloatRect ) : Boolean;
8577: Function GEqualRect( const R1, R2 : TRect ) : Boolean;
8578: Function EqualRect1( const R1, R2 : TFloatRect ) : Boolean;
8579: Procedure GInflateRect( var R : TRect; Dx, Dy : Integer);
8580: Procedure InflateRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8581: Procedure GOffsetRect( var R : TRect; Dx, Dy : Integer);
8582: Procedure OffsetRect1( var FR : TFloatRect; Dx, Dy : TFloat);
8583: Function IsRectEmpty( const R : TRect ) : Boolean;
8584: Function IsRectEmpty1( const FR : TFloatRect ) : Boolean;
8585: Function GPtInRect( const R : TRect; const P : TPoint ) : Boolean;
8586: Function PtInRect1( const R : TFloatRect; const P : TPoint ) : Boolean;
8587: Function PtInRect2( const R : TRect; const P : TFloatPoint ) : Boolean;
8588: Function PtInRect3( const R : TFloatRect; const P : TFloatPoint ) : Boolean;
8589: Function EqualRectSize( const R1, R2 : TRect ) : Boolean;
8590: Function EqualRectSize1( const R1, R2 : TFloatRect ) : Boolean;
8591: Function MessageBeep( uType : UINT ) : BOOL
8592: Function ShowCursor( bShow : BOOL ) : Integer
8593: Function SetCursorPos( X, Y : Integer ) : BOOL
8594: Function SetCursor( hCursor : HICON ) : HCURSOR
8595: Function GetCursorPos( var lpPoint : TPoint ) : BOOL
8596: //Function ClipCursor( lpRect : PRect ) : BOOL
8597: Function GetClipCursor( var lpRect : TRect ) : BOOL
8598: Function GetCursor : HCURSOR
8599: Function CreateCaret( hWnd : HWND; hBitmap : HBITMAP; nWidth, nHeight : Integer ) : BOOL
8600: Function GetCaretBlinkTime : UINT
8601: Function SetCaretBlinkTime( uMSeconds : UINT ) : BOOL
8602: Function DestroyCaret : BOOL
8603: Function HideCaret( hWnd : HWND ) : BOOL
8604: Function ShowCaret( hWnd : HWND ) : BOOL
8605: Function SetCaretPos( X, Y : Integer ) : BOOL
8606: Function GetCaretPos( var lpPoint : TPoint ) : BOOL
8607: Function ClientToScreen( hWnd : HWND; var lpPoint : TPoint ) : BOOL
8608: Function ScreenToClient( hWnd : HWND; var lpPoint : TPoint ) : BOOL
8609: Function MapWindowPoints( hWndFrom, hWndTo:HWND; var lpPoints, cPoints : UINT ) : Integer
8610: Function WindowFromPoint( Point : TPoint ) : HWND
8611: Function ChildWindowFromPoint( hWndParent : HWND; Point : TPoint ) : HWND
8612:
8613:
8614: procedure SIRegister_GR32_Math(CL: TPSPascalCompiler);
8615: begin
8616:   Function FixedFloor( A : TFixed): Integer
8617:   Function FixedCeil( A : TFixed) : Integer
8618:   Function FixedMul( A, B : TFixed) : TFixed
8619:   Function FixedDiv( A, B : TFixed) : TFixed
8620:   Function OneOver( Value : TFixed) : TFixed
8621:   Function FixedRound( A : TFixed) : Integer
8622:   Function FixedSqr( Value : TFixed) : TFixed
8623:   Function FixedSqrtLP( Value : TFixed) : TFixed
8624:   Function FixedSqrthp( Value : TFixed) : TFixed
8625:   Function FixedCombine( W, X, Y : TFixed) : TFixed
8626:   Procedure GRSinCos( const Theta : TFloat; out Sin, Cos : TFloat);
8627:   Procedure GRSinCos1( const Theta, Radius : Single; out Sin, Cos : Single);
8628:   Function GRHypot( const X, Y : TFloat) : TFloat;
8629:   Function Hypot1( const X, Y : Integer) : Integer;
8630:   Function FastSqrt( const Value : TFloat) : TFloat
8631:   Function FastSqrtBab1( const Value : TFloat) : TFloat
8632:   Function FastSqrtBab2( const Value : TFloat) : TFloat
8633:   Function FastInvSqrt( const Value : Single) : Single;
8634:   Function MulDiv( Multiplicand, Multiplier, Divisor : Integer ) : Integer
8635:   Function GRIsPowerOf2( Value : Integer) : Boolean
8636:   Function PrevPowerOf2( Value : Integer) : Integer
8637:   Function NextPowerOf2( Value : Integer) : Integer
8638:   Function Average( A, B : Integer) : Integer
8639:   Function GRSign( Value : Integer) : Integer
8640:   Function FloatMod( x, y : Double) : Double
8641: end;

```



```

8642:
8643: procedure SIRegister_GR32_LowLevel(CL: TPSPascalCompiler);
8644: begin
8645:   Function Clamp( const Value : Integer) : Integer;
8646:   Procedure GRFillWord( var X, Count : Cardinal; Value : Longword)
8647:   Function StackAlloc( Size : Integer) : Pointer
8648:   Procedure StackFree( P : Pointer)
8649:   Procedure Swap( var A, B : Pointer);
8650:   Procedure Swap1( var A, B : Integer);
8651:   Procedure Swap2( var A, B : TFixed);
8652:   Procedure Swap3( var A, B : TColor32);
8653:   Procedure TestSwap( var A, B : Integer);
8654:   Procedure TestSwap1( var A, B : TFixed);
8655:   Function TestClip( var A, B : Integer; const Size : Integer) : Boolean;
8656:   Function TestClip1( var A, B : Integer; const Start, Stop : Integer) : Boolean;
8657:   Function GRConstrain( const Value, Lo, Hi : Integer) : Integer;
8658:   Function Constrain1( const Value, Lo, Hi : Single) : Single;
8659:   Function SwapConstrain( const Value:Integer; Constrain1,Constrain2:Integer) : Integer
8660:   Function GRMin( const A, B, C : Integer) : Integer;
8661:   Function GRMax( const A, B, C : Integer) : Integer;
8662:   Function Clamp( Value, Max : Integer) : Integer;
8663:   Function Clamp1( Value, Min, Max : Integer) : Integer;
8664:   Function Wrap( Value, Max : Integer) : Integer;
8665:   Function Wrap1( Value, Min, Max : Integer) : Integer;
8666:   Function Wrap3( Value, Max : Single) : Single;;
8667:   Function WrapPow2( Value, Max : Integer) : Integer;
8668:   Function WrapPow21( Value, Min, Max : Integer) : Integer;
8669:   Function Mirror( Value, Max : Integer) : Integer;
8670:   Function Mirror1( Value, Min, Max : Integer) : Integer;
8671:   Function MirrorPow2( Value, Max : Integer) : Integer;
8672:   Function MirrorPow21( Value, Min, Max : Integer) : Integer;
8673:   Function GetOptimalWrap( Max : Integer) : TWrapProc;
8674:   Function GetOptimalWrap1( Min, Max : Integer) : TWrapProcEx;
8675:   Function GetOptimalMirror( Max : Integer) : TWrapProc;
8676:   Function GetOptimalMirror1( Min, Max : Integer) : TWrapProcEx;
8677:   Function GetWrapProc( WrapMode : TWrapMode) : TWrapProc;
8678:   Function GetWrapProc1( WrapMode : TWrapMode; Max : Integer) : TWrapProc;
8679:   Function GetWrapProcEx( WrapMode : TWrapMode) : TWrapProcEx;
8680:   Function GetWrapProcEx1( WrapMode : TWrapMode; Min, Max : Integer):TWrapProcEx;
8681:   Function Div255( Value : Cardinal) : Cardinal
8682:   Function SAR_4( Value : Integer) : Integer
8683:   Function SAR_8( Value : Integer) : Integer
8684:   Function SAR_9( Value : Integer) : Integer
8685:   Function SAR_11( Value : Integer) : Integer
8686:   Function SAR_12( Value : Integer) : Integer
8687:   Function SAR_13( Value : Integer) : Integer
8688:   Function SAR_14( Value : Integer) : Integer
8689:   Function SAR_15( Value : Integer) : Integer
8690:   Function SAR_16( Value : Integer) : Integer
8691:   Function ColorSwap( WinColor : TColor) : TColor32
8692: end;
8693:
8694: procedure SIRegister_GR32_Filters(CL: TPSPascalCompiler);
8695: begin
8696:   AddTypeS('TLogicalOperator', '( loXOR, loAND, loOR )
8697:   Procedure CopyComponents( Dst, Src : TCustomBitmap32; Components : TColor32Components);
8698:   Procedure CopyComponents1(Dst:TCustomBmap32;DstX,
      DstY:Int;Src:TCustomBmap32;SrcRect:TRect;Components:TColor32Comp;
8699:   Procedure AlphaToGrayscale( Dst, Src : TCustomBitmap32)
8700:   Procedure ColorToGrayscale( Dst, Src : TCustomBitmap32; PreserveAlpha : Boolean)
8701:   Procedure IntensityToAlpha( Dst, Src : TCustomBitmap32)
8702:   Procedure Invert( Dst, Src : TCustomBitmap32; Components : TColor32Components)
8703:   Procedure InvertRGB( Dst, Src : TCustomBitmap32)
8704:   Procedure ApplyLUT( Dst, Src : TCustomBitmap32; const LUT : TLUT8; PreserveAlpha : Boolean)
8705:   Procedure ChromaKey( ABitmap : TCustomBitmap32; TrColor : TColor32)
8706:   Function CreateBitmask( Components : TColor32Components) : TColor32
8707:   Procedure ApplyBitmask(Dst:TCustomBitmap32; DstX,DstY:Integer; Src:TCustomBitmap32; SrcRect : TRect;
      Bitmask : TColor32; LogicalOperator : TLogicalOperator);
8708:   Procedure
      ApplyBitmask1(ABitmap:TCustomBitmap32;ARect:TRect;Bitmask:TColor32;LogicalOperator:TLogicalOperator);
8709:   Procedure CheckParams( Dst, Src : TCustomBitmap32; ResizedDst : Boolean)
8710: end;
8711:
8712:
8713: procedure SIRegister_JclNTFS(CL: TPSPascalCompiler);
8714: begin
8715:   AddClassN(FindClass('TOBJECT'),'EJclNtfsError
8716:   AddTypeS('TFileCompressionState', '( fcNoCompression, fcDefaultCompression, fcLZNT1Compression )
8717:   Function NtfsGetCompression( const FileName : string; var State : Short) : Boolean;
8718:   Function NtfsGetCompression1( const FileName : string) : TFileCompressionState;
8719:   Function NtfsSetCompression( const FileName : string; const State : Short) : Boolean
8720:   Procedure NtfsSetFileCompression( const FileName : string; const State : TFileCompressionState)
8721:   Procedure NtfsSetDirectoryTreeCompression(const Directory: string; const State : TFileCompressionState)
8722:   Procedure NtfsSetDefaultFileCompression(const Directory: string; const State:TFileCompressionState)
8723:   Procedure NtfsSetPathCompression(const Path:string;const State:TFileCompressionState;Recursive:Boolean;
8724:   //AddTypeS('TNtfsAllocRanges', 'record Entries : Integer; Data : PFileAlloca'
8725:   //+ 'tedRangeBuffer; MoreData : Boolean; end
8726:   Function NtfsSetSparse( const FileName : string) : Boolean
8727:   Function NtfsZeroDataByHandle( const Handle : THandle; const First, Last : Int64) : Boolean

```

```

8728: Function NtfsZeroDataByName( const FileName : string; const First, Last : Int64 ) : Boolean
8729: //Function NtfsQueryAllocRanges(const FileName:string;Offset,Count:Int64;var
Ranges:TNTfsAllocRanges):Boolean;
8730: //Function NtfsGetAllocRangeEntry( const Ranges : TNTfsAllocRanges;
Index:Integer):TFileAllocatedRangeBuffer
8731: Function NtfsSparseStreamsSupported( const Volume : string ) : Boolean
8732: Function NtfsGetSparse( const FileName : string ) : Boolean
8733: Function NtfsDeleteReparsePoint( const FileName : string; ReparseTag : DWORD ) : Boolean
8734: Function NtfsSetReparsePoint( const FileName : string; var ReparseData, Size : Longword ) : Boolean
8735: //Function NtfsGetReparsePoint(const FileName:string; var ReparseData:TReparseGuidDataBuffer):Boolean
8736: Function NtfsGetReparseTag( const Path : string; var Tag : DWORD ) : Boolean
8737: Function NtfsReparsePointsSupported( const Volume : string ) : Boolean
8738: Function NtfsFileHasReparsePoint( const Path : string ) : Boolean
8739: Function NtfsIsFolderMountPoint( const Path : string ) : Boolean
8740: Function NtfsMountDeviceAsDrive( const Device : string; Drive : Char ) : Boolean
8741: Function NtfsMountVolume( const Volume : Char; const MountPoint : string ) : Boolean
8742: AddTypeS('TopLock', '( olExclusive, olReadOnly, olBatch, olFilter )
8743: Function NtfsOpLockAckClosePending( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8744: Function NtfsOpLockBreakAckNo2( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8745: Function NtfsOpLockBreakAcknowledge( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8746: Function NtfsOpLockBreakNotify( Handle : THandle; Overlapped : TOverlapped ) : Boolean
8747: Function NtfsRequestOpLock( Handle : THandle; Kind : TopLock; Overlapped : TOverlapped ) : Boolean
8748: Function NtfsCreateJunctionPoint( const Source, Destination : string ) : Boolean
8749: Function NtfsDeleteJunctionPoint( const Source : string ) : Boolean
8750: Function NtfsGetJunctionPointDestination( const Source : string; var Destination : string ) : Boolean
8751: AddTypeS('TStreamId', '( siInvalid, siStandard, siExtendedAttribute, siSec'
8752: + 'urity, siAlternate, siHardLink, siProperty, siObjectIdentifier, siReparsePoints, siSparseFile )
8753: AddTypeS('TStreamIds', 'set of TStreamId
8754: AddTypeS('TInternalFindStreamData', 'record FileHandle : THandle; Context '
8755: + ': __Pointer; StreamIds : TStreamIds; end
8756: AddTypeS('TFindStreamData', 'record internal : TInternalFindStreamData; At'
8757: + 'tributes : DWORD; StreamID : TStreamId; Name : WideString; Size : Int64; end
8758: Function NtfsFindFirstStream(const FileName:string;StreamIds:TStreamIds;var Data:TFindStreamData):Boolean;
8759: Function NtfsFindNextStream( var Data : TFindStreamData ) : Boolean
8760: Function NtfsFindStreamClose( var Data : TFindStreamData ) : Boolean
8761: Function NtfsCreateHardLink( const LinkFileName, ExistingFileName : string ) : Boolean
8762: AddTypeS('TNTfsHardLinkInfo', 'record LinkCount : Cardinal; FileIndex : Int64; end
8763: Function NtfsGetHardLinkInfo( const FileName : string; var Info : TNTfsHardLinkInfo ) : Boolean
8764: Function NtfsFindHardLinks(const Path:string;const FileIndexHigh,FileIndexLow:Cardinal;const
List:TStrings):Bool;
8765: Function NtfsDeleteHardLinks( const FileName : string ) : Boolean
8766: Function JclAppInstances : TJclAppInstances;
8767: Function JclAppInstances1( const UniqueAppIdGuidStr : string ) : TJclAppInstances;
8768: Function ReadMessageCheck( var Message: TMessage;const IgnoredOriginatorWnd: HWND ) : TJclAppInstDataKind
8769: Procedure ReadMessageData( const Message : TMessage; var Data : __Pointer; var Size : Integer )
8770: Procedure ReadMessageString( const Message : TMessage; var S : string )
8771: Procedure ReadMessageStrings( const Message : TMessage; const Strings : TStrings )
8772:
8773:
8774: (*-----*)
8775: procedure SIRegister_JclGraphics(CL: TPSPascalCompiler);
8776: begin
8777: FindClass('TOBJECT'),'EJclGraphicsError
8778: TDynDynIntegerArrayArray', 'array of TDynIntegerArray
8779: TDynPointArray', 'array of TPoint
8780: TDynDynPointArrayArray', 'array of TDynPointArray
8781: TPointF', 'record X : Single; Y : Single; end
8782: TDynPointArrayF', 'array of TPointF
8783: TDrawMode2', '( dmOpaque, dmBlend )
8784: TStretchFilter2', '( sfNearest, sfLinear, sfSpline )
8785: TConversionKind', '( ckRed, ckGreen, ckBlue, ckAlpha, ckUniformRGB, ckWeightedRGB )
8786: TResamplingFilter', '( rfBox, rfTriangle, rfHermite, rfBell, rfSpline, rfLanczos3, rfMitchell )
8787: TMatrix3d', 'record array[0..2,0..2] of extended end
8788: TDynDynPointArrayArrayF', 'array of TDynPointArrayF
8789: TScanLine', 'array of Integer
8790: TScanLines', 'array of TScanLine
8791: TColorChannel', '( ccRed, ccGreen, ccBlue, ccAlpha )
8792: TGradientDirection', '( gdVertical, gdHorizontal )
8793: TPolyFillMode', '( fmAlternate, fmWinding )
8794: TJclRegionCombineOperator', '( coAnd, coDiff, coOr, coXor )
8795: TJclRegionBitmapMode', '( rmInclude, rmExclude )
8796: TJclRegionKind', '( rkNull, rkSimple, rkComplex, rkError )
8797: SIRegister_TJclDesktopCanvas(CL);
8798: FindClass('TOBJECT'),'TJclRegion
8799: SIRegister_TJclRegionInfo(CL);
8800: SIRegister_TJclRegion(CL);
8801: SIRegister_TJclThreadPersistent(CL);
8802: SIRegister_TJclCustomMap(CL);
8803: SIRegister_TJclBitmap32(CL);
8804: SIRegister_TJclByteMap(CL);
8805: SIRegister_TJclTransformation(CL);
8806: SIRegister_TJclLinearTransformation(CL);
8807: Procedure Stretch(NewWidth,
NewHeight:Card;Filter:TResamplingFilter;Radius:Single;Source:TGraphic;Target:TBitmap);
8808: Procedure Stretch1(NewWidth,NewHeight:Cardinal;Filter:TResamplingFilter;Radius:Single;Bitmap:TBitmap);
8809: Procedure DrawBitmap( DC : HDC; Bitmap : HBitmap; X, Y, Width, Height : Integer )
8810: Function GetAntialiasedBitmap( const Bitmap : TBitmap ) : TBitmap
8811: Procedure BitmapToJpeg( const FileName : string )
8812: Procedure JPegToBitmap( const FileName : string )

```

```

8813: Function ExtractIconCount( const FileName : string ) : Integer
8814: Function BitmapToIconJ( Bitmap : HBITMAP; cx, cy : Integer ) : HICON
8815: Function IconToBitmapJ( Icon : HICON ) : HBITMAP
8816: Procedure BlockTransfer( Dst : TJclBitmap32; DstX : Integer; DstY : Integer; Src : TJclBitmap32; SrcRect : TRect; CombineOp : TDrawMode )
8817: Procedure StretchTransfer(Dst:TJclBitmap32;
DstRect:TRect; Src:TJclBitmap32; SrcRect:TRect; StretchFilter:TStretchFilter; CombineOp : TDrawMode )
8818: Procedure Transform( Dst, Src : TJclBitmap32; SrcRect : TRect; Transformation : TJclTransformation )
8819: Procedure SetBorderTransparent( ABitmap : TJclBitmap32; ARect : TRect )
8820: Function FillGradient( DC : HDC; ARect : TRect; ColorCount : Integer; StartColor, EndColor : TColor; ADirection : TGradientDirection ) : Boolean;
8821: Function CreateRegionFromBitmap(Bitmap: TBitmap; RegionColor:TColor;
RegionBitmapMode:TJclRegionBitmapMode): HRGN
8822: Procedure ScreenShot( bm : TBitmap; Left, Top, Width, Height : Integer; Window : HWND );
8823: Procedure ScreenShot1( bm : TBitmap );
8824: Procedure PolyLineTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32 )
8825: Procedure PolyLineAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32 )
8826: Procedure PolyLineFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32 )
8827: Procedure PolygonTS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32 )
8828: Procedure PolygonAS( Bitmap : TJclBitmap32; const Points : TDynPointArray; Color : TColor32 )
8829: Procedure PolygonFS( Bitmap : TJclBitmap32; const Points : TDynPointArrayF; Color : TColor32 )
8830: Procedure PolyPolygonTS(Bitmap:TJclBitmap32; const Points:TDynDynPointArrayArray; Color:TColor32);
8831: Procedure PolyPolygonAS(Bitmap:TJclBitmap32; const Points:TDynDynPointArrayArray; Color:TColor32);
8832: Procedure PolyPolygonFS(Bitmap:TJclBitmap32; const Points:TDynDynPointArrayArrayF; Color:TColor32);
8833: Procedure AlphaToGrayscale( Dst, Src : TJclBitmap32 )
8834: Procedure IntensityToAlpha( Dst, Src : TJclBitmap32 )
8835: Procedure Invert( Dst, Src : TJclBitmap32 )
8836: Procedure InvertRGB( Dst, Src : TJclBitmap32 )
8837: Procedure ColorToGrayscale( Dst, Src : TJclBitmap32 )
8838: Procedure ApplyLUT( Dst, Src : TJclBitmap32; const LUT : TLUT8 )
8839: Procedure SetGamma( Gamma : Single )
8840: end;
8841:
8842: (*-----*)
8843: procedure SIRegister_JclSynch(CL: TPSPascalCompiler);
8844: begin
8845:   Function LockedAdd( var Target : Integer; Value : Integer ) : Integer
8846:   Function LockedCompareExchange( var Target : Integer; Exch, Comp : Integer ) : Integer;
8847:   Function LockedCompareExchangel( var Target : __Pointer; Exch, Comp : __Pointer ) : Pointer;
8848:   Function LockedDec( var Target : Integer ) : Integer
8849:   Function LockedExchange( var Target : Integer; Value : Integer ) : Integer
8850:   Function LockedExchangeAdd( var Target : Integer; Value : Integer ) : Integer
8851:   Function LockedExchangeDec( var Target : Integer ) : Integer
8852:   Function LockedExchangeInc( var Target : Integer ) : Integer
8853:   Function LockedExchangeSub( var Target : Integer; Value : Integer ) : Integer
8854:   Function LockedInc( var Target : Integer ) : Integer
8855:   Function LockedSub( var Target : Integer; Value : Integer ) : Integer
8856:   TJclWaitResult, '( wrAbandoned, wrError, wrIoCompletion, wrSignaled, wrTimeout )
8857:   SIRegister_TJclDispatcherObject(CL);
8858:   Function WaitForMultipleObjects(const Objects:array of
TJclDispatcherObject; WaitAll:Bool; Timeout:Cardinal):Cardinal;
8859:   Function WaitAlertableForMultipleObjects(const Objects : array of TJclDispatcherObject; WaitAll:Bool;
Timeout : Cardinal):Cardinal
8860:   SIRegister_TJclCriticalSection(CL);
8861:   SIRegister_TJclCriticalSectionEx(CL);
8862:   SIRegister_TJclEvent(CL);
8863:   SIRegister_TJclWaitableTimer(CL);
8864:   SIRegister_TJclSemaphore(CL);
8865:   SIRegister_TJclMutex(CL);
8866:   POptexSharedInfo, '^TOptexSharedInfo // will not work
8867:   TOptexSharedInfo, 'record SpinCount:Int; LockCount: Int; ThreadId:Longword; RecursionCount:Int; end
8868:   SIRegister_TJclOptex(CL);
8869:   TMrewPreferred, '( mpReaders, mpWriters, mpEqual )
8870:   TMrewThreadInfo, 'record ThreadId : Longword; RecursionCount: Integer; Reader : Boolean; end
8871:   TMrewThreadInfoArray, 'array of TMrewThreadInfo
8872:   SIRegister_TJclMultiReadExclusiveWrite(CL);
8873:   PMetSectSharedInfo, '^TMetSectSharedInfo // will not work
8874:   TMetSectSharedInfo, 'record Initialized : LongBool; SpinLock : '
+Longint; ThreadsWaiting : Longint; AvailableCount : Longint; MaximumCount : Longint; end
8875:   PMeteredSection, '^TMeteredSection // will not work
8876:   TMeteredSection, 'record Event : THandle; FileMap : THandle; SharedInfo : PMetSectSharedInfo; end
8877:   SIRegister_TJclMeteredSection(CL);
8878:   TEventInfo, 'record EventType : Longint; Signaled : LongBool; end
8879:   TMutexInfo, 'record SignalState : Longint; Owned : Boolean; Abandoned : Boolean; end
8880:   TSemaphoreCounts, 'record CurrentCount : Longint; MaximumCount : Longint; end
8881:   TTimerInfo, 'record Remaining : TLargeInteger; Signaled : LongBool; end
8882:   Function QueryCriticalSection( CS : TJclCriticalSection; var Info : TRTLCriticalSection ) : Boolean
8883:   Function QueryEvent( Handle : THandle; var Info : TEventInfo ) : Boolean
8884:   Function QueryMutex( Handle : THandle; var Info : TMutexInfo ) : Boolean
8885:   Function QuerySemaphore( Handle : THandle; var Info : TSemaphoreCounts ) : Boolean
8886:   Function QueryTimer( Handle : THandle; var Info : TTimerInfo ) : Boolean
8887:   FindClass('TOBJECT'), EJclWin32HandleObjectError
8888:   FindClass('TOBJECT'), EJclDispatcherObjectError
8889:   FindClass('TOBJECT'), EJclCriticalSectionError
8890:   FindClass('TOBJECT'), EJclEventError
8891:   FindClass('TOBJECT'), EJclWaitableTimerError
8892:   FindClass('TOBJECT'), EJclSemaphoreError
8893:   FindClass('TOBJECT'), EJclMutexError
8894:   FindClass('TOBJECT'), EJclMeteredSectionError
8895:   FindClass('TOBJECT'), EJclMeteredSectionError

```

```

8896: end;
8897:
8898:
8899: //*****unit uPSI_mORMotReport;
8900: Procedure SetCurrentPrinterAsDefault
8901: Function CurrentPrinterName : string
8902: Function mCurrentPrinterPaperSize : string
8903: Procedure UseDefaultPrinter
8904:
8905: procedure SIRegisterTSTREAM(C1: TPSPascalCompiler);
8906: begin
8907:   with FindClass('TOBJECT', 'TStream') do begin
8908:     IsAbstract := True;
8909:     //RegisterMethod('Function Read( var Buffer, Count : Longint) : Longint
8910:     // RegisterMethod('Function Write( const Buffer, Count : Longint) : Longint
8911:     function Read(Buffer:String;Count:LongInt):LongInt
8912:     function Write(Buffer:String;Count:LongInt):LongInt
8913:     procedure ReadAB(Buffer: TByteArray;Count:LongInt)
8914:     procedure WriteAB(Buffer: TByteArray;Count:LongInt)
8915:     procedure ReadABD(Buffer: TByteDynArray;Count:LongInt)
8916:     procedure WriteABD(Buffer: TByteDynArray;Count:LongInt)
8917:     procedure ReadAC(Buffer: TCharArray;Count:LongInt)
8918:     procedure WriteAC(Buffer: TCharArray;Count:LongInt)
8919:     procedure ReadACD(Buffer: TCharDynArray;Count:LongInt)
8920:     procedure WriteACD(Buffer: TCharDynArray;Count:LongInt)
8921:
8922:     function Seek(Offset:LongInt;Origin:Word):LongInt
8923:     procedure ReadBuffer(Buffer:String;Count:LongInt)
8924:     procedure WriteBuffer(Buffer:String;Count:LongInt)
8925:     procedure ReadBufferAB(Buffer: TByteArray;Count:LongInt)
8926:     procedure WriteBufferAB(Buffer: TByteArray;Count:LongInt)
8927:     procedure ReadBufferABD(Buffer: TByteDynArray;Count:LongInt)
8928:     procedure WriteBufferABD(Buffer: TByteDynArray;Count:LongInt)
8929:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
8930:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
8931:     procedure ReadBufferAWD(Buffer: TWordDynArray;Count:LongInt)
8932:     procedure WriteBufferAWD(Buffer: TWordDynArray;Count:LongInt)
8933:     procedure ReadBufferAW(Buffer: TWordArray;Count:LongInt)
8934:     procedure WriteBufferAW(Buffer: TWordArray;Count:LongInt)
8935:     procedure ReadBufferAC(Buffer: TCharArray;Count:LongInt)
8936:     procedure WriteBufferAC(Buffer: TCharArray;Count:LongInt)
8937:     procedure ReadBufferACD(Buffer: TCharDynArray;Count:LongInt)
8938:     procedure WriteBufferACD(Buffer: TCharDynArray;Count:LongInt)
8939:
8940:     procedure ReadBufferP(Buffer: PChar;Count:LongInt)
8941:     procedure WriteBufferP(Buffer: PChar;Count:LongInt)
8942:     procedure ReadBufferO(Buffer: TObject;Count:LongInt)';
8943:     procedure WriteBufferO(Buffer: TObject;Count:LongInt)';
8944:     //READBUFFERAC
8945:     function InstanceSize: Longint
8946:     Procedure FixupResourceHeader( FixupInfo : Integer)
8947:     Procedure ReadResHeader
8948:
8949:     {$IFDEF DELPHI4UP}
8950:     function CopyFrom(Source:TStream;Count:Int64):LongInt
8951:     {$ELSE}
8952:     function CopyFrom(Source:TStream;Count:Integer):LongInt
8953:     {$ENDIF}
8954:     RegisterProperty('Position', 'LongInt', iptrw);
8955:     RegisterProperty('Size', 'LongInt', iptrw);
8956:   end;
8957: end;
8958:
8959:
8960: { *****
8961:   Unit DMATH - Interface for DMATH.DLL
8962:   ***** }
8963: // see more docs/dmath_manual.pdf
8964:
8965: Function InitEval : Integer
8966: Procedure SetVariable( VarName : Char; Value : Float)
8967: Procedure SetFunction( FuncName : String; Wrapper : TWrapper)
8968: Function Eval( ExpressionString : String) : Float
8969:
8970: unit dmath; //types are in built, others are external in DLL
8971: interface
8972: {$IFDEF DELPHI}
8973: uses
8974:   StdCtrls, Graphics;
8975: {$ENDIF}
8976: { -----
8977:   Types and constants
8978: ----- }
8979: {$i types.inc}
8980: { -----
8981:   Error handling
8982: ----- }
8983: procedure SetErrCode(ErrCode : Integer); external 'dmath';
8984: { Sets the error code }

```



```

8985: function DefaultVal(ErrCode : Integer; DefVal : Float) : Float; external 'dmath';
8986: { Sets error code and default function value }
8987: function MathErr : Integer; external 'dmath';
8988: { Returns the error code }
8989: { -----
8990:   Dynamic arrays
8991: ----- }
8992: procedure SetAutoInit(AutoInit : Boolean); external 'dmath';
8993: { Sets the auto-initialization of arrays }
8994: procedure DimVector(var V : TVector; Ub : Integer); external 'dmath';
8995: { Creates floating point vector V[0..Ub] }
8996: procedure DimIntVector(var V : TIntVector; Ub : Integer); external 'dmath';
8997: { Creates integer vector V[0..Ub] }
8998: procedure DimCompVector(var V : TCompVector; Ub : Integer); external 'dmath';
8999: { Creates complex vector V[0..Ub] }
9000: procedure DimBoolVector(var V : TBoolVector; Ub : Integer); external 'dmath';
9001: { Creates boolean vector V[0..Ub] }
9002: procedure DimStrVector(var V : TStrVector; Ub : Integer); external 'dmath';
9003: { Creates string vector V[0..Ub] }
9004: procedure DimMatrix(var A : TMatrix; Ub1, Ub2 : Integer); external 'dmath';
9005: { Creates floating point matrix A[0..Ub1, 0..Ub2] }
9006: procedure DimIntMatrix(var A : TIntMatrix; Ub1, Ub2 : Integer); external 'dmath';
9007: { Creates integer matrix A[0..Ub1, 0..Ub2] }
9008: procedure DimCompMatrix(var A : TCompMatrix; Ub1, Ub2 : Integer); external 'dmath';
9009: { Creates complex matrix A[0..Ub1, 0..Ub2] }
9010: procedure DimBoolMatrix(var A : TBoolMatrix; Ub1, Ub2 : Integer); external 'dmath';
9011: { Creates boolean matrix A[0..Ub1, 0..Ub2] }
9012: procedure DimStrMatrix(var A : TStrMatrix; Ub1, Ub2 : Integer); external 'dmath';
9013: { Creates string matrix A[0..Ub1, 0..Ub2] }
9014: { -----
9015:   Minimum, maximum, sign and exchange
9016: ----- }
9017: function FMin(X, Y : Float) : Float; external 'dmath';
9018: { Minimum of 2 reals }
9019: function FMax(X, Y : Float) : Float; external 'dmath';
9020: { Maximum of 2 reals }
9021: function IMin(X, Y : Integer) : Integer; external 'dmath';
9022: { Minimum of 2 integers }
9023: function IMax(X, Y : Integer) : Integer; external 'dmath';
9024: { Maximum of 2 integers }
9025: function Sgn(X : Float) : Integer; external 'dmath';
9026: { Sign (returns 1 if X = 0) }
9027: function Sgn0(X : Float) : Integer; external 'dmath';
9028: { Sign (returns 0 if X = 0) }
9029: function DSgn(A, B : Float) : Float; external 'dmath';
9030: { Sgn(B) * |A| }
9031: procedure FSwap(var X, Y : Float); external 'dmath';
9032: { Exchange 2 reals }
9033: procedure ISwap(var X, Y : Integer); external 'dmath';
9034: { Exchange 2 integers }
9035: { -----
9036:   Rounding functions
9037: ----- }
9038: function RoundN(X : Float; N : Integer) : Float; external 'dmath';
9039: { Rounds X to N decimal places }
9040: function Ceil(X : Float) : Integer; external 'dmath';
9041: { Ceiling function }
9042: function Floor(X : Float) : Integer; external 'dmath';
9043: { Floor function }
9044: { -----
9045:   Logarithms, exponentials and power
9046: ----- }
9047: function Expo(X : Float) : Float; external 'dmath';
9048: { Exponential }
9049: function Exp2(X : Float) : Float; external 'dmath';
9050: { 2^X }
9051: function Exp10(X : Float) : Float; external 'dmath';
9052: { 10^X }
9053: function Log(X : Float) : Float; external 'dmath';
9054: { Natural log }
9055: function Log2(X : Float) : Float; external 'dmath';
9056: { Log, base 2 }
9057: function Log10(X : Float) : Float; external 'dmath';
9058: { Decimal log }
9059: function LogA(X, A : Float) : Float; external 'dmath';
9060: { Log, base A }
9061: function IntPower(X : Float; N : Integer) : Float; external 'dmath';
9062: { X^N }
9063: function Power(X, Y : Float) : Float; external 'dmath';
9064: { X^Y, X >= 0 }
9065: { -----
9066:   Trigonometric functions
9067: ----- }
9068: function Pythag(X, Y : Float) : Float; external 'dmath';
9069: { Sqrt(X^2 + Y^2) }
9070: function FixAngle(Theta : Float) : Float; external 'dmath';
9071: { Set Theta in -Pi..Pi }
9072: function Tan(X : Float) : Float; external 'dmath';
9073: { Tangent }

```

```

9074: function ArcSin(X : Float) : Float; external 'dmath';
9075: { Arc sinus }
9076: function ArcCos(X : Float) : Float; external 'dmath';
9077: { Arc cosinus }
9078: function ArcTan2(Y, X : Float) : Float; external 'dmath';
9079: { Angle (Ox, OM) with M(X,Y) }
9080: { -----
9081:   Hyperbolic functions
9082: ----- }
9083: function Sinh(X : Float) : Float; external 'dmath';
9084: { Hyperbolic sine }
9085: function Cosh(X : Float) : Float; external 'dmath';
9086: { Hyperbolic cosine }
9087: function Tanh(X : Float) : Float; external 'dmath';
9088: { Hyperbolic tangent }
9089: function ArcSinh(X : Float) : Float; external 'dmath';
9090: { Inverse hyperbolic sine }
9091: function ArcCosh(X : Float) : Float; external 'dmath';
9092: { Inverse hyperbolic cosine }
9093: function ArcTanh(X : Float) : Float; external 'dmath';
9094: { Inverse hyperbolic tangent }
9095: procedure SinhCosh(X : Float; var SinhX, CoshX : Float); external 'dmath';
9096: { Sinh & Cosh }
9097: { -----
9098:   Gamma function and related functions
9099: ----- }
9100: function Fact(N : Integer) : Float; external 'dmath';
9101: { Factorial }
9102: function SgnGamma(X : Float) : Integer; external 'dmath';
9103: { Sign of Gamma function }
9104: function Gamma(X : Float) : Float; external 'dmath';
9105: { Gamma function }
9106: function LnGamma(X : Float) : Float; external 'dmath';
9107: { Logarithm of Gamma function }
9108: function Stirling(X : Float) : Float; external 'dmath';
9109: { Stirling's formula for the Gamma function }
9110: function StirLog(X : Float) : Float; external 'dmath';
9111: { Approximate Ln(Gamma) by Stirling's formula, for X >= 13 }
9112: function DiGamma(X : Float) : Float; external 'dmath';
9113: { Digamma function }
9114: function TriGamma(X : Float) : Float; external 'dmath';
9115: { Trigamma function }
9116: function IGamma(A, X : Float) : Float; external 'dmath';
9117: { Incomplete Gamma function }
9118: function JGamma(A, X : Float) : Float; external 'dmath';
9119: { Complement of incomplete Gamma function }
9120: function InvGamma(A, P : Float) : Float; external 'dmath';
9121: { Inverse of incomplete Gamma function }
9122: function Erf(X : Float) : Float; external 'dmath';
9123: { Error function }
9124: function Erfc(X : Float) : Float; external 'dmath';
9125: { Complement of error function }
9126: { -----
9127:   Beta function and related functions
9128: ----- }
9129: function Beta(X, Y : Float) : Float; external 'dmath';
9130: { Beta function }
9131: function IBeta(A, B, X : Float) : Float; external 'dmath';
9132: { Incomplete Beta function }
9133: function InvBeta(A, B, Y : Float) : Float; external 'dmath';
9134: { Inverse of incomplete Beta function }
9135: { -----
9136:   Lambert's function
9137: ----- }
9138: function LambertW(X : Float; UBranch, Offset : Boolean) : Float; external 'dmath';
9139: { -----
9140:   Binomial distribution
9141: ----- }
9142: function Binomial(N, K : Integer) : Float; external 'dmath';
9143: { Binomial coefficient C(N,K) }
9144: function PBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9145: { Probability of binomial distribution }
9146: function FBinom(N : Integer; P : Float; K : Integer) : Float; external 'dmath';
9147: { Cumulative probability for binomial distrib. }
9148: { -----
9149:   Poisson distribution
9150: ----- }
9151: function PPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9152: { Probability of Poisson distribution }
9153: function FPoisson(Mu : Float; K : Integer) : Float; external 'dmath';
9154: { Cumulative probability for Poisson distrib. }
9155: { -----
9156:   Exponential distribution
9157: ----- }
9158: function DExpo(A, X : Float) : Float; external 'dmath';
9159: { Density of exponential distribution with parameter A }
9160: function FExpo(A, X : Float) : Float; external 'dmath';
9161: { Cumulative probability function for exponential dist. with parameter A }
9162: { -----

```

```

9163:   Standard normal distribution
9164:   ----- }
9165: function DNorm(X : Float) : Float; external 'dmath';
9166: { Density of standard normal distribution }
9167: function FNorm(X : Float) : Float; external 'dmath';
9168: { Cumulative probability for standard normal distrib. }
9169: function PNorm(X : Float) : Float; external 'dmath';
9170: { Prob(|U| > X) for standard normal distrib. }
9171: function InvNorm(P : Float) : Float; external 'dmath';
9172: { Inverse of standard normal distribution }
9173: { ----- }
9174:   Student's distribution
9175:   ----- }
9176: function DStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9177: { Density of Student distribution with Nu d.o.f. }
9178: function FStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9179: { Cumulative probability for Student distrib. with Nu d.o.f. }
9180: function PStudent(Nu : Integer; X : Float) : Float; external 'dmath';
9181: { Prob(|t| > X) for Student distrib. with Nu d.o.f. }
9182: function InvStudent(Nu : Integer; P : Float) : Float; external 'dmath';
9183: { Inverse of Student's t-distribution function }
9184: { ----- }
9185:   Khi-2 distribution
9186:   ----- }
9187: function DKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9188: { Density of Khi-2 distribution with Nu d.o.f. }
9189: function FKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9190: { Cumulative prob. for Khi-2 distrib. with Nu d.o.f. }
9191: function PKhi2(Nu : Integer; X : Float) : Float; external 'dmath';
9192: { Prob(Khi2 > X) for Khi-2 distrib. with Nu d.o.f. }
9193: function InvKhi2(Nu : Integer; P : Float) : Float; external 'dmath';
9194: { Inverse of Khi-2 distribution function }
9195: { ----- }
9196:   Fisher-Snedecor distribution
9197:   ----- }
9198: function DSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9199: { Density of Fisher-Snedecor distribution with Nu1 and Nu2 d.o.f. }
9200: function FSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9201: { Cumulative prob. for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9202: function PSnedecor(Nu1, Nu2 : Integer; X : Float) : Float; external 'dmath';
9203: { Prob(F > X) for Fisher-Snedecor distrib. with Nu1 and Nu2 d.o.f. }
9204: function InvSnedecor(Nu1, Nu2 : Integer; P : Float) : Float; external 'dmath';
9205: { Inverse of Snedecor's F-distribution function }
9206: { ----- }
9207:   Beta distribution
9208:   ----- }
9209: function DBeta(A, B, X : Float) : Float; external 'dmath';
9210: { Density of Beta distribution with parameters A and B }
9211: function FBeta(A, B, X : Float) : Float; external 'dmath';
9212: { Cumulative probability for Beta distrib. with param. A and B }
9213: { ----- }
9214:   Gamma distribution
9215:   ----- }
9216: function DGamma(A, B, X : Float) : Float; external 'dmath';
9217: { Density of Gamma distribution with parameters A and B }
9218: function FGamma(A, B, X : Float) : Float; external 'dmath';
9219: { Cumulative probability for Gamma distrib. with param. A and B }
9220: { ----- }
9221:   Expression evaluation
9222:   ----- }
9223: function InitEval : Integer; external 'dmath';
9224: { Initializes built-in functions and returns their number }
9225: function Eval(ExpressionString : String) : Float; external 'dmath';
9226: { Evaluates an expression at run-time }
9227: procedure SetVariable(VarName : Char; Value : Float); external 'dmath';
9228: { Assigns a value to a variable }
9229: procedure SetFunction(FuncName : String; Wrapper : TWrapper); external 'dmath';
9230: { Adds a function to the parser }
9231: { ----- }
9232:   Matrices and linear equations
9233:   ----- }
9234: procedure GaussJordan(A : TMatrix;
9235:   Lb, Ubl, Ub2 : Integer;
9236:   var Det : Float); external 'dmath';
9237: { Transforms a matrix according to the Gauss-Jordan method }
9238: procedure LinEq(A : TMatrix;
9239:   B : TVector;
9240:   Lb, Ub : Integer;
9241:   var Det : Float); external 'dmath';
9242: { Solves a linear system according to the Gauss-Jordan method }
9243: procedure Cholesky(A, L : TMatrix; Lb, Ub : Integer); external 'dmath';
9244: { Cholesky factorization of a positive definite symmetric matrix }
9245: procedure LU_Decom(A : TMatrix; Lb, Ub : Integer); external 'dmath';
9246: { LU decomposition }
9247: procedure LU_Solve(A : TMatrix;
9248:   B : TVector;
9249:   Lb, Ub : Integer;
9250:   X : TVector); external 'dmath';
9251: { Solution of linear system from LU decomposition }

```

```

9252: procedure QR_Decomp(A      : TMatrix;
9253:                     Lb, Ub1, Ub2 : Integer;
9254:                     R      : TMatrix); external 'dmath';
9255: { QR decomposition }
9256: procedure QR_Solve(Q, R      : TMatrix;
9257:                   B      : TVector;
9258:                   Lb, Ub1, Ub2 : Integer;
9259:                   X      : TVector); external 'dmath';
9260: { Solution of linear system from QR decomposition }
9261: procedure SV_Decomp(A      : TMatrix;
9262:                    Lb, Ub1, Ub2 : Integer;
9263:                    S      : TVector;
9264:                    V      : TMatrix); external 'dmath';
9265: { Singular value decomposition }
9266: procedure SV_SetZero(S      : TVector;
9267:                    Lb, Ub : Integer;
9268:                    Tol   : Float); external 'dmath';
9269: { Set lowest singular values to zero }
9270: procedure SV_Solve(U      : TMatrix;
9271:                   S      : TVector;
9272:                   V      : TMatrix;
9273:                   B      : TVector;
9274:                   Lb, Ub1, Ub2 : Integer;
9275:                   X      : TVector); external 'dmath';
9276: { Solution of linear system from SVD }
9277: procedure SV_Approx(U      : TMatrix;
9278:                    S      : TVector;
9279:                    V      : TMatrix;
9280:                    Lb, Ub1, Ub2 : Integer;
9281:                    A      : TMatrix); external 'dmath';
9282: { Matrix approximation from SVD }
9283: procedure EigenVals(A      : TMatrix;
9284:                    Lb, Ub : Integer;
9285:                    Lambda : TCompVector); external 'dmath';
9286: { Eigenvalues of a general square matrix }
9287: procedure EigenVect(A      : TMatrix;
9288:                    Lb, Ub : Integer;
9289:                    Lambda : TCompVector;
9290:                    V      : TMatrix); external 'dmath';
9291: { Eigenvalues and eigenvectors of a general square matrix }
9292: procedure EigenSym(A      : TMatrix;
9293:                   Lb, Ub : Integer;
9294:                   Lambda : TVector;
9295:                   V      : TMatrix); external 'dmath';
9296: { Eigenvalues and eigenvectors of a symmetric matrix (SVD method) }
9297: procedure Jacobi(A      : TMatrix;
9298:                 Lb, Ub, MaxIter : Integer;
9299:                 Tol           : Float;
9300:                 Lambda       : TVector;
9301:                 V            : TMatrix); external 'dmath';
9302: { Eigenvalues and eigenvectors of a symmetric matrix (Jacobi method) }
9303: { -----
9304: Optimization
9305: ----- }
9306: procedure MinBrack(Func      : TFunc;
9307:                   var A, B, C, Fa, Fb, Fc : Float); external 'dmath';
9308: { Brackets a minimum of a function }
9309: procedure GoldSearch(Func      : TFunc;
9310:                     A, B      : Float;
9311:                     MaxIter   : Integer;
9312:                     Tol       : Float;
9313:                     var Xmin, Ymin : Float); external 'dmath';
9314: { Minimization of a function of one variable (golden search) }
9315: procedure LinMin(Func      : TFuncNVar;
9316:                 X, DeltaX : TVector;
9317:                 Lb, Ub    : Integer;
9318:                 var R      : Float;
9319:                 MaxIter   : Integer;
9320:                 Tol       : Float;
9321:                 var F_min : Float); external 'dmath';
9322: { Minimization of a function of several variables along a line }
9323: procedure Newton(Func      : TFuncNVar;
9324:                 HessGrad  : THessGrad;
9325:                 X          : TVector;
9326:                 Lb, Ub    : Integer;
9327:                 MaxIter   : Integer;
9328:                 Tol       : Float;
9329:                 var F_min : Float;
9330:                 G         : TVector;
9331:                 H_inv     : TMatrix;
9332:                 var Det   : Float); external 'dmath';
9333: { Minimization of a function of several variables (Newton's method) }
9334: procedure SaveNewton(FileName : string); external 'dmath';
9335: { Save Newton iterations in a file }
9336: procedure Marquardt(Func      : TFuncNVar;
9337:                    HessGrad  : THessGrad;
9338:                    X          : TVector;
9339:                    Lb, Ub    : Integer;
9340:                    MaxIter   : Integer;

```



```

9341:         Tol      : Float;
9342:         var F_min : Float;
9343:         G         : TVector;
9344:         H_inv     : TMatrix;
9345:         var Det   : Float); external 'dmath';
9346: { Minimization of a function of several variables (Marquardt's method) }
9347: procedure SaveMarquardt(FileName : string); external 'dmath';
9348: { Save Marquardt iterations in a file }
9349: procedure BFGS(Func      : TFuncNVar;
9350:               Gradient  : TGradient;
9351:               X         : TVector;
9352:               Lb, Ub    : Integer;
9353:               MaxIter   : Integer;
9354:               Tol       : Float;
9355:               var F_min : Float;
9356:               G         : TVector;
9357:               H_inv     : TMatrix); external 'dmath';
9358: { Minimization of a function of several variables (BFGS method) }
9359: procedure SaveBFGS(FileName : string); external 'dmath';
9360: { Save BFGS iterations in a file }
9361: procedure Simplex(Func      : TFuncNVar;
9362:                  X         : TVector;
9363:                  Lb, Ub    : Integer;
9364:                  MaxIter   : Integer;
9365:                  Tol       : Float;
9366:                  var F_min : Float); external 'dmath';
9367: { Minimization of a function of several variables (Simplex) }
9368: procedure SaveSimplex(FileName : string); external 'dmath';
9369: { Save Simplex iterations in a file }
9370: { -----
9371:   Nonlinear equations
9372:   ----- }
9373: procedure RootBrack(Func      : TFunc;
9374:                    var X, Y, FX, FY : Float); external 'dmath';
9375: { Brackets a root of function Func between X and Y }
9376: procedure Bisect(Func      : TFunc;
9377:                 var X, Y : Float;
9378:                 MaxIter  : Integer;
9379:                 Tol      : Float;
9380:                 var F    : Float); external 'dmath';
9381: { Bisection method }
9382: procedure Secant(Func      : TFunc;
9383:                 var X, Y : Float;
9384:                 MaxIter  : Integer;
9385:                 Tol      : Float;
9386:                 var F    : Float); external 'dmath';
9387: { Secant method }
9388: procedure NewtEq(Func, Deriv : TFunc;
9389:                 var X      : Float;
9390:                 MaxIter    : Integer;
9391:                 Tol        : Float;
9392:                 var F      : Float); external 'dmath';
9393: { Newton-Raphson method for a single nonlinear equation }
9394: procedure NewtEqs(Equations : TEquations;
9395:                  Jacobian   : TJacobian;
9396:                  X, F       : TVector;
9397:                  Lb, Ub    : Integer;
9398:                  MaxIter   : Integer;
9399:                  Tol       : Float); external 'dmath';
9400: { Newton-Raphson method for a system of nonlinear equations }
9401: procedure Broyden(Equations : TEquations;
9402:                  X, F       : TVector;
9403:                  Lb, Ub    : Integer;
9404:                  MaxIter   : Integer;
9405:                  Tol       : Float); external 'dmath';
9406: { Broyden's method for a system of nonlinear equations }
9407: { -----
9408:   Polynomials and rational fractions
9409:   ----- }
9410: function Poly(X      : Float;
9411:              Coef    : TVector;
9412:              Deg      : Integer) : Float; external 'dmath';
9413: { Evaluates a polynomial }
9414: function RFrac(X      : Float;
9415:               Coef    : TVector;
9416:               Deg1, Deg2 : Integer) : Float; external 'dmath';
9417: { Evaluates a rational fraction }
9418: function RootPol1(A, B : Float;
9419:                  var X : Float) : Integer; external 'dmath';
9420: { Solves the linear equation A + B * X = 0 }
9421: function RootPol2(Coef : TVector;
9422:                  Z      : TCompVector) : Integer; external 'dmath';
9423: { Solves a quadratic equation }
9424: function RootPol3(Coef : TVector;
9425:                  Z      : TCompVector) : Integer; external 'dmath';
9426: { Solves a cubic equation }
9427: function RootPol4(Coef : TVector;
9428:                  Z      : TCompVector) : Integer; external 'dmath';
9429: { Solves a quartic equation }

```

```

9430: function RootPol(Coef : TVector;
9431:                  Deg  : Integer;
9432:                  Z     : TCompVector) : Integer; external 'dmath';
9433: { Solves a polynomial equation }
9434: function SetRealRoots(Deg : Integer;
9435:                      Z     : TCompVector;
9436:                      Tol   : Float) : Integer; external 'dmath';
9437: { Set the imaginary part of a root to zero }
9438: procedure SortRoots(Deg : Integer;
9439:                    Z     : TCompVector); external 'dmath';
9440: { Sorts the roots of a polynomial }
9441: { -----
9442:   Numerical integration and differential equations
9443:   ----- }
9444: function TrapInt(X, Y : TVector; N : Integer) : Float; external 'dmath';
9445: { Integration by trapezoidal rule }
9446: function GausLeg(Func : TFunc; A, B : Float) : Float; external 'dmath';
9447: { Integral from A to B }
9448: function GausLeg0(Func : TFunc; B : Float) : Float; external 'dmath';
9449: { Integral from 0 to B }
9450: function Convolve(Func1, Func2 : TFunc; T : Float) : Float; external 'dmath';
9451: { Convolution product at time T }
9452: procedure ConvTrap(Func1, Func2 : TFunc; T, Y : TVector; N : Integer); external 'dmath';
9453: { Convolution by trapezoidal rule }
9454: procedure RKF45(F      : TDiffEqs;
9455:                Negn    : Integer;
9456:                Y, Yp    : TVector;
9457:                var T     : Float;
9458:                Tout, RelErr, AbsErr : Float;
9459:                var Flag  : Integer); external 'dmath';
9460: { Integration of a system of differential equations }
9461: { -----
9462:   Fast Fourier Transform
9463:   ----- }
9464: procedure FFT(NumSamples : Integer;
9465:              InArray, OutArray : TCompVector); external 'dmath';
9466: { Fast Fourier Transform }
9467: procedure IFFT(NumSamples : Integer;
9468:              InArray, OutArray : TCompVector); external 'dmath';
9469: { Inverse Fast Fourier Transform }
9470: procedure FFT_Integer(NumSamples : Integer;
9471:                      RealIn, ImagIn : TIntVector;
9472:                      OutArray : TCompVector); external 'dmath';
9473: { Fast Fourier Transform for integer data }
9474: procedure FFT_Integer_Cleanup; external 'dmath';
9475: { Clear memory after a call to FFT_Integer }
9476: procedure CalcFrequency(NumSamples,
9477:                       FrequencyIndex : Integer;
9478:                       InArray : TCompVector;
9479:                       var FFT : Complex); external 'dmath';
9480: { Direct computation of Fourier transform }
9481: { -----
9482:   Random numbers
9483:   ----- }
9484: procedure SetRNG(RNG : RNG_Type); external 'dmath';
9485: { Select generator }
9486: procedure InitGen(Seed : RNG_IntType); external 'dmath';
9487: { Initialize generator }
9488: function IRanGen : RNG_IntType; external 'dmath';
9489: { 32-bit random integer in [-2^31 .. 2^31 - 1] }
9490: function IRanGen31 : RNG_IntType; external 'dmath';
9491: { 31-bit random integer in [0 .. 2^31 - 1] }
9492: function RanGen1 : Float; external 'dmath';
9493: { 32-bit random real in [0,1] }
9494: function RanGen2 : Float; external 'dmath';
9495: { 32-bit random real in [0,1] }
9496: function RanGen3 : Float; external 'dmath';
9497: { 32-bit random real in [0,1] }
9498: function RanGen53 : Float; external 'dmath';
9499: { 53-bit random real in [0,1] }
9500: procedure InitMWC(Seed : RNG_IntType); external 'dmath';
9501: { Initializes the 'Multiply with carry' random number generator }
9502: function IRanMWC : RNG_IntType; external 'dmath';
9503: { Returns a 32 bit random number in [-2^31 ; 2^31-1] }
9504: procedure InitMT(Seed : RNG_IntType); external 'dmath';
9505: { Initializes Mersenne Twister generator with a seed }
9506: procedure InitMTbyArray(InitKey : array of RNG_LongType;
9507:                       KeyLength : Word); external 'dmath';
9508: { Initialize MT generator with an array InitKey[0..(KeyLength - 1)] }
9509: function IRanMT : RNG_IntType; external 'dmath';
9510: { Random integer from MT generator }
9511: procedure InitUVAGbyString(KeyPhrase : string); external 'dmath';
9512: { Initializes the UVAG generator with a string }
9513: procedure InitUVAG(Seed : RNG_IntType); external 'dmath';
9514: { Initializes the UVAG generator with an integer }
9515: function IRanUVAG : RNG_IntType; external 'dmath';
9516: { Random integer from UVAG generator }
9517: function RanGaussStd : Float; external 'dmath';
9518: { Random number from standard normal distribution }

```

```

9519: function RanGauss(Mu, Sigma : Float) : Float; external 'dmath';
9520: { Random number from normal distrib. with mean Mu and S. D. Sigma }
9521: procedure RanMult(M      : TVector; L      : TMatrix;
9522:                  Lb, Ub : Integer;
9523:                  X      : TVector); external 'dmath';
9524: { Random vector from multinormal distribution (correlated) }
9525: procedure RanMultIndep(M, S : TVector;
9526:                      Lb, Ub : Integer;
9527:                      X      : TVector); external 'dmath';
9528: { Random vector from multinormal distribution (uncorrelated) }
9529: procedure InitMHPParams(NCycles, MaxSim, SavedSim : Integer); external 'dmath';
9530: { Initializes Metropolis-Hastings parameters }
9531: procedure GetMHPParams(var NCycles, MaxSim, SavedSim: Integer); external 'dmath';
9532: { Returns Metropolis-Hastings parameters }
9533: procedure Hastings(Func      : TFuncNVar;
9534:                   T          : Float;
9535:                   X          : TVector;
9536:                   V          : TMatrix;
9537:                   Lb, Ub     : Integer;
9538:                   Xmat       : TMatrix;
9539:                   X_min      : TVector;
9540:                   var F_min   : Float); external 'dmath';
9541: { Simulation of a probability density function by Metropolis-Hastings }
9542: procedure InitSAParams(NT, NS, NCycles : Integer; RT : Float); external 'dmath';
9543: { Initializes Simulated Annealing parameters }
9544: procedure SA_CreateLogFile(FileName : String); external 'dmath';
9545: { Initializes log file }
9546: procedure SimAnn(Func      : TFuncNVar;
9547:                 X, Xmin, Xmax : TVector;
9548:                 Lb, Ub      : Integer;
9549:                 var F_min    : Float); external 'dmath';
9550: { Minimization of a function of several var. by simulated annealing }
9551: procedure InitGAParams(NP, NG : Integer; SR, MR, HR : Float); external 'dmath';
9552: { Initializes Genetic Algorithm parameters }
9553: procedure GA_CreateLogFile(FileName : String); external 'dmath';
9554: { Initializes log file }
9555: procedure GenAlg(Func      : TFuncNVar;
9556:                 X, Xmin, Xmax : TVector;
9557:                 Lb, Ub      : Integer;
9558:                 var F_min    : Float); external 'dmath';
9559: { Minimization of a function of several var. by genetic algorithm }
9560: { -----
9561:   Statistics
9562:   ----- }
9563: function Mean(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9564: { Mean of sample X }
9565: function Min(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9566: { Minimum of sample X }
9567: function Max(X : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9568: { Maximum of sample X }
9569: function Median(X : TVector; Lb, Ub : Integer; Sorted : Boolean) : Float; external 'dmath';
9570: { Median of sample X }
9571: function StDev(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9572: { Standard deviation estimated from sample X }
9573: function StDevP(X : TVector; Lb, Ub : Integer; M : Float) : Float; external 'dmath';
9574: { Standard deviation of population }
9575: function Correl(X, Y : TVector; Lb, Ub : Integer) : Float; external 'dmath';
9576: { Correlation coefficient }
9577: function Skewness(X : TVector; Lb, Ub : Integer; M, Sigma: Float) : Float; external 'dmath';
9578: { Skewness of sample X }
9579: function Kurtosis(X : TVector; Lb, Ub : Integer; M, Sigma: Float) : Float; external 'dmath';
9580: { Kurtosis of sample X }
9581: procedure QSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9582: { Quick sort (ascending order) }
9583: procedure DQSort(X : TVector; Lb, Ub : Integer); external 'dmath';
9584: { Quick sort (descending order) }
9585: procedure Interval(X1, X2      : Float;
9586:                  MinDiv, MaxDiv : Integer;
9587:                  var Min, Max, Step : Float); external 'dmath';
9588: { Determines an interval for a set of values }
9589: procedure AutoScale(X : TVector; Lb, Ub : Integer; Scale : TScale;
9590:                   var XMin, XMax, XStep : Float); external 'dmath';
9591: { Finds an appropriate scale for plotting the data in X[Lb..Ub] }
9592: procedure StudIndep(N1, N2      : Integer;
9593:                   M1, M2, S1, S2 : Float;
9594:                   var T          : Float;
9595:                   var DoF        : Integer); external 'dmath';
9596: { Student t-test for independent samples }
9597: procedure StudPaired(X, Y      : TVector;
9598:                   Lb, Ub      : Integer;
9599:                   var T        : Float;
9600:                   var DoF      : Integer); external 'dmath';
9601: { Student t-test for paired samples }
9602: procedure AnOVal(Ns      : Integer;
9603:                 N         : TIntVector;
9604:                 M, S      : TVector;
9605:                 var V_f, V_r, F : Float;
9606:                 var DoF_f, DoF_r : Integer); external 'dmath';
9607: { One-way analysis of variance }

```

```

9608: procedure AnOVA2(NA, NB, Nobs : Integer;
9609:                  M, S          : TMatrix;
9610:                  V, F          : TVector;
9611:                  DoF           : TIntVector); external 'dmath';
9612: { Two-way analysis of variance }
9613: procedure Snedecor(N1, N2      : Integer;
9614:                  S1, S2      : Float;
9615:                  var F       : Float;
9616:                  var DoF1, DoF2 : Integer); external 'dmath';
9617: { Snedecor's F-test (comparison of two variances) }
9618: procedure Bartlett(Ns         : Integer;
9619:                  N          : TIntVector;
9620:                  S          : TVector;
9621:                  var Khi2    : Float;
9622:                  var DoF     : Integer); external 'dmath';
9623: { Bartlett's test (comparison of several variances) }
9624: procedure Mann_Whitney(N1, N2 : Integer;
9625:                  X1, X2      : TVector;
9626:                  var U, Eps   : Float); external 'dmath';
9627: { Mann-Whitney test }
9628: procedure Wilcoxon(X, Y      : TVector;
9629:                  Lb, Ub     : Integer;
9630:                  var Ndiff   : Integer;
9631:                  var T, Eps   : Float); external 'dmath';
9632: { Wilcoxon test }
9633: procedure Kruskal_Wallis(Ns      : Integer;
9634:                  N            : TIntVector;
9635:                  X            : TMatrix;
9636:                  var H        : Float;
9637:                  var DoF      : Integer); external 'dmath';
9638: { Kruskal-Wallis test }
9639: procedure Khi2_Conform(N_cls     : Integer;
9640:                  N_estim        : Integer;
9641:                  Obs            : TIntVector;
9642:                  Calc           : TVector;
9643:                  var Khi2       : Float;
9644:                  var DoF        : Integer); external 'dmath';
9645: { Khi-2 test for conformity }
9646: procedure Khi2_Indep(N_lin      : Integer;
9647:                  N_col         : Integer;
9648:                  Obs           : TIntMatrix;
9649:                  var Khi2      : Float;
9650:                  var DoF       : Integer); external 'dmath';
9651: { Khi-2 test for independence }
9652: procedure Woolf_Conform(N_cls    : Integer;
9653:                  N_estim         : Integer;
9654:                  Obs            : TIntVector;
9655:                  Calc           : TVector;
9656:                  var G          : Float;
9657:                  var DoF        : Integer); external 'dmath';
9658: { Woolf's test for conformity }
9659: procedure Woolf_Indep(N_lin     : Integer;
9660:                  N_col          : Integer;
9661:                  Obs            : TIntMatrix;
9662:                  var G          : Float;
9663:                  var DoF        : Integer); external 'dmath';
9664: { Woolf's test for independence }
9665: procedure DimStatClassVector(var C : TStatClassVector;
9666:                  Ub             : Integer); external 'dmath';
9667: { Allocates an array of statistical classes: C[0..Ub] }
9668: procedure Distrib(X          : TVector;
9669:                  Lb, Ub     : Integer;
9670:                  A, B, H    : Float;
9671:                  C          : TStatClassVector); external 'dmath';
9672: { Distributes an array X[Lb..Ub] into statistical classes }
9673: { -----
9674:   Linear / polynomial regression
9675:   ----- }
9676: procedure LinFit(X, Y      : TVector;
9677:                  Lb, Ub     : Integer;
9678:                  B          : TVector;
9679:                  V          : TMatrix); external 'dmath';
9680: { Linear regression : Y = B(0) + B(1) * X }
9681: procedure WLinFit(X, Y, S : TVector;
9682:                  Lb, Ub   : Integer;
9683:                  B        : TVector;
9684:                  V        : TMatrix); external 'dmath';
9685: { Weighted linear regression : Y = B(0) + B(1) * X }
9686: procedure SVDLinFit(X, Y      : TVector;
9687:                  Lb, Ub     : Integer;
9688:                  SVDTol     : Float;
9689:                  B          : TVector;
9690:                  V          : TMatrix); external 'dmath';
9691: { Unweighted linear regression by singular value decomposition }
9692: procedure WSVDLinFit(X, Y, S : TVector;
9693:                  Lb, Ub     : Integer;
9694:                  SVDTol     : Float;
9695:                  B          : TVector;
9696:                  V          : TMatrix); external 'dmath';

```



```

9697: { Weighted linear regression by singular value decomposition }
9698: procedure MulFit(X      : TMatrix;
9699:                 Y      : TVector;
9700:                 Lb, Ub, Nvar : Integer;
9701:                 ConstTerm : Boolean;
9702:                 B        : TVector;
9703:                 V        : TMatrix); external 'dmath';
9704: { Multiple linear regression by Gauss-Jordan method }
9705: procedure WMulFit(X      : TMatrix;
9706:                  Y, S    : TVector;
9707:                  Lb, Ub, Nvar : Integer;
9708:                  ConstTerm : Boolean;
9709:                  B        : TVector;
9710:                  V        : TMatrix); external 'dmath';
9711: { Weighted multiple linear regression by Gauss-Jordan method }
9712: procedure SVDFit(X      : TMatrix;
9713:                  Y      : TVector;
9714:                  Lb, Ub, Nvar : Integer;
9715:                  ConstTerm : Boolean;
9716:                  SVDTol    : Float;
9717:                  B        : TVector;
9718:                  V        : TMatrix); external 'dmath';
9719: { Multiple linear regression by singular value decomposition }
9720: procedure WSVDFit(X      : TMatrix;
9721:                  Y, S    : TVector;
9722:                  Lb, Ub, Nvar : Integer;
9723:                  ConstTerm : Boolean;
9724:                  SVDTol    : Float;
9725:                  B        : TVector;
9726:                  V        : TMatrix); external 'dmath';
9727: { Weighted multiple linear regression by singular value decomposition }
9728: procedure PolFit(X, Y      : TVector;
9729:                 Lb, Ub, Deg : Integer;
9730:                 B          : TVector;
9731:                 V          : TMatrix); external 'dmath';
9732: { Polynomial regression by Gauss-Jordan method }
9733: procedure WPolFit(X, Y, S : TVector;
9734:                  Lb, Ub, Deg : Integer;
9735:                  B          : TVector;
9736:                  V          : TMatrix); external 'dmath';
9737: { Weighted polynomial regression by Gauss-Jordan method }
9738: procedure SVDPolFit(X, Y      : TVector;
9739:                    Lb, Ub, Deg : Integer;
9740:                    SVDTol    : Float;
9741:                    B          : TVector;
9742:                    V          : TMatrix); external 'dmath';
9743: { Unweighted polynomial regression by singular value decomposition }
9744: procedure WSVDPolFit(X, Y, S : TVector;
9745:                    Lb, Ub, Deg : Integer;
9746:                    SVDTol    : Float;
9747:                    B          : TVector;
9748:                    V          : TMatrix); external 'dmath';
9749: { Weighted polynomial regression by singular value decomposition }
9750: procedure RegTest(Y, Ycalc : TVector;
9751:                  LbY, UbY : Integer;
9752:                  V        : TMatrix;
9753:                  LbV, UbV : Integer;
9754:                  var Test : TRegTest); external 'dmath';
9755: { Test of unweighted regression }
9756: procedure WRegTest(Y, Ycalc, S : TVector;
9757:                   LbY, UbY    : Integer;
9758:                   V          : TMatrix;
9759:                   LbV, UbV    : Integer;
9760:                   var Test    : TRegTest); external 'dmath';
9761: { Test of weighted regression }
9762: { -----
9763:   Nonlinear regression
9764:   ----- }
9765: procedure SetOptAlgo(Algo : TOptAlgo); external 'dmath';
9766: { Sets the optimization algorithm for nonlinear regression }
9767: function GetOptAlgo : TOptAlgo; external 'dmath';
9768: { Returns the optimization algorithm }
9769: procedure SetMaxParam(N : Byte); external 'dmath';
9770: { Sets the maximum number of regression parameters for nonlinear regression }
9771: function GetMaxParam : Byte; external 'dmath';
9772: { Returns the maximum number of regression parameters for nonlinear regression }
9773: procedure SetParamBounds(I : Byte; ParamMin, ParamMax : Float); external 'dmath';
9774: { Sets the bounds on the I-th regression parameter }
9775: procedure GetParamBounds(I : Byte; var ParamMin, ParamMax:Float); external 'dmath';
9776: { Returns the bounds on the I-th regression parameter }
9777: procedure NLFit(RegFunc : TRegFunc;
9778:                DerivProc : TDerivProc;
9779:                X, Y      : TVector;
9780:                Lb, Ub    : Integer;
9781:                MaxIter   : Integer;
9782:                Tol       : Float;
9783:                B         : TVector;
9784:                FirstPar,
9785:                LastPar   : Integer;

```

```

9786:         V          : TMatrix); external 'dmath';
9787: { Unweighted nonlinear regression }
9788: procedure WNLFit(RegFunc : TRegFunc;
9789:                 DerivProc : TDerivProc;
9790:                 X, Y, S : TVector;
9791:                 Lb, Ub : Integer;
9792:                 MaxIter : Integer;
9793:                 Tol : Float;
9794:                 B : TVector;
9795:                 FirstPar,
9796:                 LastPar : Integer;
9797:                 V : TMatrix); external 'dmath';
9798: { Weighted nonlinear regression }
9799: procedure SetMCFile(FileName : String); external 'dmath';
9800: { Set file for saving MCMC simulations }
9801: procedure SimFit(RegFunc : TRegFunc;
9802:                 X, Y : TVector;
9803:                 Lb, Ub : Integer;
9804:                 B : TVector;
9805:                 FirstPar,
9806:                 LastPar : Integer;
9807:                 V : TMatrix); external 'dmath';
9808: { Simulation of unweighted nonlinear regression by MCMC }
9809: procedure WSimFit(RegFunc : TRegFunc;
9810:                 X, Y, S : TVector;
9811:                 Lb, Ub : Integer;
9812:                 B : TVector;
9813:                 FirstPar,
9814:                 LastPar : Integer;
9815:                 V : TMatrix); external 'dmath';
9816: { Simulation of weighted nonlinear regression by MCMC }
9817: { -----
9818:   Nonlinear regression models
9819: ----- }
9820: procedure FracFit(X, Y : TVector;
9821:                 Lb, Ub : Integer;
9822:                 Deg1, Deg2 : Integer;
9823:                 ConsTerm : Boolean;
9824:                 MaxIter : Integer;
9825:                 Tol : Float;
9826:                 B : TVector;
9827:                 V : TMatrix); external 'dmath';
9828: { Unweighted fit of rational fraction }
9829: procedure WFracFit(X, Y, S : TVector;
9830:                 Lb, Ub : Integer;
9831:                 Deg1, Deg2 : Integer;
9832:                 ConsTerm : Boolean;
9833:                 MaxIter : Integer;
9834:                 Tol : Float;
9835:                 B : TVector;
9836:                 V : TMatrix); external 'dmath';
9837: { Weighted fit of rational fraction }
9838:
9839: function FracFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9840: { Returns the value of the rational fraction at point X }
9841: procedure ExpFit(X, Y : TVector;
9842:                 Lb, Ub, Nexp : Integer;
9843:                 ConsTerm : Boolean;
9844:                 MaxIter : Integer;
9845:                 Tol : Float;
9846:                 B : TVector;
9847:                 V : TMatrix); external 'dmath';
9848: { Unweighted fit of sum of exponentials }
9849: procedure WExpFit(X, Y, S : TVector;
9850:                 Lb, Ub, Nexp : Integer;
9851:                 ConsTerm : Boolean;
9852:                 MaxIter : Integer;
9853:                 Tol : Float;
9854:                 B : TVector;
9855:                 V : TMatrix); external 'dmath';
9856: { Weighted fit of sum of exponentials }
9857: function ExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9858: { Returns the value of the regression function at point X }
9859: procedure IncExpFit(X, Y : TVector;
9860:                 Lb, Ub : Integer;
9861:                 ConsTerm : Boolean;
9862:                 MaxIter : Integer;
9863:                 Tol : Float;
9864:                 B : TVector;
9865:                 V : TMatrix); external 'dmath';
9866: { Unweighted fit of model of increasing exponential }
9867: procedure WIncExpFit(X, Y, S : TVector;
9868:                 Lb, Ub : Integer;
9869:                 ConsTerm : Boolean;
9870:                 MaxIter : Integer;
9871:                 Tol : Float;
9872:                 B : TVector;
9873:                 V : TMatrix); external 'dmath';
9874: { Weighted fit of increasing exponential }

```

```

9875: function IncExpFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9876: { Returns the value of the regression function at point X }
9877: procedure ExpLinFit(X, Y : TVector;
9878:                   Lb, Ub : Integer;
9879:                   MaxIter : Integer;
9880:                   Tol : Float;
9881:                   B : TVector;
9882:                   V : TMatrix); external 'dmath';
9883: { Unweighted fit of the "exponential + linear" model }
9884: procedure WExpLinFit(X, Y, S : TVector;
9885:                   Lb, Ub : Integer;
9886:                   MaxIter : Integer;
9887:                   Tol : Float;
9888:                   B : TVector;
9889:                   V : TMatrix); external 'dmath';
9890: { Weighted fit of the "exponential + linear" model }
9891:
9892: function ExpLinFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9893: { Returns the value of the regression function at point X }
9894: procedure MichFit(X, Y : TVector;
9895:                  Lb, Ub : Integer;
9896:                  MaxIter : Integer;
9897:                  Tol : Float;
9898:                  B : TVector;
9899:                  V : TMatrix); external 'dmath';
9900: { Unweighted fit of Michaelis equation }
9901: procedure WMichFit(X, Y, S : TVector;
9902:                  Lb, Ub : Integer;
9903:                  MaxIter : Integer;
9904:                  Tol : Float;
9905:                  B : TVector;
9906:                  V : TMatrix); external 'dmath';
9907: { Weighted fit of Michaelis equation }
9908: function MichFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9909: { Returns the value of the Michaelis equation at point X }
9910: procedure MintFit(X, Y : TVector;
9911:                  Lb, Ub : Integer;
9912:                  MintVar : TMintVar;
9913:                  Fit_S0 : Boolean;
9914:                  MaxIter : Integer;
9915:                  Tol : Float;
9916:                  B : TVector;
9917:                  V : TMatrix); external 'dmath';
9918: { Unweighted fit of the integrated Michaelis equation }
9919: procedure WMintFit(X, Y, S : TVector;
9920:                  Lb, Ub : Integer;
9921:                  MintVar : TMintVar;
9922:                  Fit_S0 : Boolean;
9923:                  MaxIter : Integer;
9924:                  Tol : Float;
9925:                  B : TVector;
9926:                  V : TMatrix); external 'dmath';
9927: { Weighted fit of the integrated Michaelis equation }
9928: function MintFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9929: { Returns the value of the integrated Michaelis equation at point X }
9930: procedure HillFit(X, Y : TVector;
9931:                  Lb, Ub : Integer;
9932:                  MaxIter : Integer;
9933:                  Tol : Float;
9934:                  B : TVector;
9935:                  V : TMatrix); external 'dmath';
9936: { Unweighted fit of Hill equation }
9937: procedure WHillFit(X, Y, S : TVector;
9938:                  Lb, Ub : Integer;
9939:                  MaxIter : Integer;
9940:                  Tol : Float;
9941:                  B : TVector;
9942:                  V : TMatrix); external 'dmath';
9943: { Weighted fit of Hill equation }
9944: function HillFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9945: { Returns the value of the Hill equation at point X }
9946: procedure LogiFit(X, Y : TVector;
9947:                  Lb, Ub : Integer;
9948:                  ConsTerm : Boolean;
9949:                  General : Boolean;
9950:                  MaxIter : Integer;
9951:                  Tol : Float;
9952:                  B : TVector;
9953:                  V : TMatrix); external 'dmath';
9954: { Unweighted fit of logistic function }
9955: procedure WLogiFit(X, Y, S : TVector;
9956:                  Lb, Ub : Integer;
9957:                  ConsTerm : Boolean;
9958:                  General : Boolean;
9959:                  MaxIter : Integer;
9960:                  Tol : Float;
9961:                  B : TVector;
9962:                  V : TMatrix); external 'dmath';
9963: { Weighted fit of logistic function }

```

```

9964: function LogiFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9965: { Returns the value of the logistic function at point X }
9966: procedure PKFit(X, Y      : TVector;
9967:                Lb, Ub    : Integer;
9968:                MaxIter   : Integer;
9969:                Tol       : Float;
9970:                B         : TVector;
9971:                V         : TMatrix); external 'dmath';
9972: { Unweighted fit of the acid-base titration curve }
9973: procedure WPKFit(X, Y, S : TVector;
9974:                 Lb, Ub  : Integer;
9975:                 MaxIter : Integer;
9976:                 Tol     : Float;
9977:                 B       : TVector;
9978:                 V       : TMatrix); external 'dmath';
9979: { Weighted fit of the acid-base titration curve }
9980: function PKFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9981: { Returns the value of the acid-base titration function at point X }
9982: procedure PowFit(X, Y      : TVector;
9983:                 Lb, Ub    : Integer;
9984:                 MaxIter   : Integer;
9985:                 Tol       : Float;
9986:                 B         : TVector;
9987:                 V         : TMatrix); external 'dmath';
9988: { Unweighted fit of power function }
9989: procedure WPowFit(X, Y, S : TVector;
9990:                  Lb, Ub  : Integer;
9991:                  MaxIter : Integer;
9992:                  Tol     : Float;
9993:                  B       : TVector;
9994:                  V       : TMatrix); external 'dmath';
9995: { Weighted fit of power function }
9996:
9997: function PowFit_Func(X : Float; B : TVector) : Float; external 'dmath';
9998: { Returns the value of the power function at point X }
9999: procedure GammaFit(X, Y      : TVector;
10000:                   Lb, Ub    : Integer;
10001:                   MaxIter   : Integer;
10002:                   Tol       : Float;
10003:                   B         : TVector;
10004:                   V         : TMatrix); external 'dmath';
10005: { Unweighted fit of gamma distribution function }
10006: procedure WGammaFit(X, Y, S : TVector;
10007:                    Lb, Ub  : Integer;
10008:                    MaxIter : Integer;
10009:                    Tol     : Float;
10010:                    B       : TVector;
10011:                    V       : TMatrix); external 'dmath';
10012: { Weighted fit of gamma distribution function }
10013: function GammaFit_Func(X : Float; B : TVector) : Float; external 'dmath';
10014: { Returns the value of the gamma distribution function at point X }
10015: { -----
10016:   Principal component analysis
10017:   ----- }
10018: procedure VecMean(X      : TMatrix;
10019:                  Lb, Ub, Nvar : Integer;
10020:                  M          : TVector); external 'dmath';
10021: { Computes the mean vector M from matrix X }
10022: procedure VecSD(X      : TMatrix;
10023:                 Lb, Ub, Nvar : Integer;
10024:                 M, S        : TVector); external 'dmath';
10025: { Computes the vector of standard deviations S from matrix X }
10026: procedure MatVarCov(X      : TMatrix;
10027:                    Lb, Ub, Nvar : Integer;
10028:                    M          : TVector;
10029:                    V          : TMatrix); external 'dmath';
10030: { Computes the variance-covariance matrix V from matrix X }
10031: procedure MatCorrel(V      : TMatrix;
10032:                   Nvar : Integer;
10033:                   R      : TMatrix); external 'dmath';
10034: { Computes the correlation matrix R from the var-cov matrix V }
10035: procedure PCA(R      : TMatrix;
10036:              Nvar : Integer;
10037:              Lambda : TVector;
10038:              C, Rc   : TMatrix); external 'dmath';
10039: { Performs a principal component analysis of the correlation matrix R }
10040: procedure ScaleVar(X      : TMatrix;
10041:                   Lb, Ub, Nvar : Integer;
10042:                   M, S        : TVector;
10043:                   Z          : TMatrix); external 'dmath';
10044: { Scales a set of variables by subtracting means and dividing by SD's }
10045: procedure PrinFac(Z      : TMatrix;
10046:                  Lb, Ub, Nvar : Integer;
10047:                  C, F        : TMatrix); external 'dmath';
10048: { Computes principal factors }
10049: { -----
10050:   Strings
10051:   ----- }
10052: function LTrim(S : String) : String; external 'dmath';

```



```

10053: { Removes leading blanks }
10054: function RTrim(S : String) : String; external 'dmath';
10055: { Removes trailing blanks }
10056: function Trim(S : String) : String; external 'dmath';
10057: { Removes leading and trailing blanks }
10058: function StrChar(N : Byte; C : Char) : String; external 'dmath';
10059: { Returns a string made of character C repeated N times }
10060: function RFill(S : String; L : Byte) : String; external 'dmath';
10061: { Completes string S with trailing blanks for a total length L }
10062: function LFill(S : String; L : Byte) : String; external 'dmath';
10063: { Completes string S with leading blanks for a total length L }
10064: function CFill(S : String; L : Byte) : String; external 'dmath';
10065: { Centers string S on a total length L }
10066: function Replace(S : String; C1, C2 : Char) : String; external 'dmath';
10067: { Replaces in string S all the occurrences of C1 by C2 }
10068: function Extract(S : String; var Index : Byte; Delim : Char) : String; external 'dmath';
10069: { Extracts a field from a string }
10070: procedure Parse(S : String; Delim:Char; Field:TStrVector; var N:Byte); external 'dmath';
10071: { Parses a string into its constitutive fields }
10072: procedure SetFormat(NumLength,MaxDec:Integer;FloatPoint,NSZero:Bool); external 'dmath';
10073: { Sets the numeric format }
10074: function FloatStr(X : Float) : String; external 'dmath';
10075: { Converts a real to a string according to the numeric format }
10076: function IntStr(N : LongInt) : String; external 'dmath';
10077: { Converts an integer to a string }
10078: function CompStr(Z : Complex) : String; external 'dmath';
10079: { Converts a complex number to a string }
10080: {$IFDEF DELPHI}
10081: function StrDec(S : String) : String; external 'dmath';
10082: { Set decimal separator to the symbol defined in SysUtils }
10083: function IsNumeric(var S : String; var X : Float) : Boolean; external 'dmath';
10084: { Test if a string represents a number and returns it in X }
10085: function ReadNumFromEdit(Edit : TEdit) : Float; external 'dmath';
10086: { Reads a floating point number from an Edit control }
10087: procedure WriteNumToFile(var F : Text; X : Float); external 'dmath';
10088: { Writes a floating point number in a text file }
10089: {$ENDIF}
10090: { -----
10091:   BGI / Delphi graphics
10092: ----- }
10093: function InitGraphics
10094: {$IFDEF DELPHI}
10095: (Width, Height : Integer) : Boolean;
10096: {$ELSE}
10097: (Pilot, Mode : Integer; BGIPath : String) : Boolean; {$ENDIF} external 'dmath';
10098: { Enters graphic mode }
10099: procedure SetWindow({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10100:   X1, X2, Y1,Y2 : Integer; GraphBorder:Boolean); external 'dmath';
10101: { Sets the graphic window }
10102: procedure SetOxScale(Scale : TScale;
10103:   OxMin, OxMax, OxStep : Float); external 'dmath';
10104: { Sets the scale on the Ox axis }
10105: procedure SetOyScale(Scale : TScale;
10106:   OyMin, OyMax, OyStep : Float); external 'dmath';
10107: { Sets the scale on the Oy axis }
10108: procedure GetOxScale(var Scale : TScale;
10109:   var OxMin, OxMax, OxStep : Float); external 'dmath';
10110: { Returns the scale on the Ox axis }
10111: procedure GetOyScale(var Scale : TScale;
10112:   var OyMin, OyMax, OyStep : Float); external 'dmath';
10113: { Returns the scale on the Oy axis }
10114: procedure SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10115: procedure SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
10116: procedure SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10117: function GetGraphTitle : String; external 'dmath'; { Returns the title for the graph }
10118: function GetOxTitle : String; external 'dmath'; { Returns the title for the Ox axis }
10119: function GetOyTitle : String; external 'dmath'; { Returns the title for the Oy axis }
10120: {$IFDEF DELPHI}
10121: procedure SetTitleFont(FontIndex, Width, Height : Integer); external 'dmath';
10122: { Sets the font for the main graph title }
10123: procedure SetOxFont(FontIndex, Width, Height : Integer); external 'dmath';
10124: { Sets the font for the Ox axis (title and labels) }
10125: procedure SetOyFont(FontIndex, Width, Height : Integer); external 'dmath';
10126: { Sets the font for the Oy axis (title and labels) }
10127: procedure SetLgdFont(FontIndex, Width, Height : Integer); external 'dmath';
10128: { Sets the font for the legends }
10129: procedure SetClipping(Clip : Boolean); external 'dmath';
10130: { Determines whether drawings are clipped at the current viewport
10131:   boundaries, according to the value of the Boolean parameter Clip }
10132: {$ENDIF}
10133: procedure PlotOxAxis({$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10134: { Plots the horizontal axis }
10135: procedure PlotOyAxis({$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10136: { Plots the vertical axis }
10137: procedure PlotGrid({$IFDEF DELPHI}Canvas:TCanvas;{$ENDIF} Grid:TGrid); external 'dmath';
10138: { Plots a grid on the graph }
10139: procedure WriteGraphTitle({$IFDEF DELPHI}(Canvas : TCanvas){$ENDIF}; external 'dmath';
10140: { Writes the title of the graph }
10141: procedure SetMaxCurv(NCurv : Byte); external 'dmath';

```

```

10142: { Sets the maximum number of curves and re-initializes their parameters }
10143: procedure SetPointParam
10144: {$IFDEF DELPHI}
10145: (CurvIndex, Symbol, Size : Integer; Color : TColor);
10146: {$ELSE}
10147: (CurvIndex, Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10148: { Sets the point parameters for curve # CurvIndex }
10149: procedure SetLineParam
10150: {$IFDEF DELPHI}
10151: (CurvIndex : Integer; Style : TPenStyle; Width : Integer; Color : TColor);
10152: {$ELSE}
10153: (CurvIndex, Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10154: { Sets the line parameters for curve # CurvIndex }
10155: procedure SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10156: { Sets the legend for curve # CurvIndex }
10157: procedure SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10158: { Sets the step for curve # CurvIndex }
10159: function GetMaxCurv : Byte; external 'dmath'; { Returns the maximum number of curves }
10160: procedure GetPointParam
10161: {$IFDEF DELPHI}
10162: (CurvIndex : Integer; var Symbol, Size : Integer; var Color : TColor);
10163: {$ELSE}
10164: (CurvIndex : Integer; var Symbol, Size, Color : Integer); {$ENDIF} external 'dmath';
10165: { Returns the point parameters for curve # CurvIndex }
10166: procedure GetLineParam
10167: {$IFDEF DELPHI}
10168: (CurvIndex : Integer; var Style : TPenStyle; var Width : Integer; var Color : TColor);
10169: {$ELSE}
10170: (CurvIndex : Integer; var Style, Width, Color : Integer); {$ENDIF} external 'dmath';
10171: { Returns the line parameters for curve # CurvIndex }
10172: function GetCurvLegend(CurvIndex : Integer) : String; external 'dmath';
10173: { Returns the legend for curve # CurvIndex }
10174: function GetCurvStep(CurvIndex : Integer) : Integer; external 'dmath';
10175: { Returns the step for curve # CurvIndex }
10176: {$IFDEF DELPHI}
10177: procedure PlotPoint(Canvas : TCanvas;
10178: X, Y : Float; CurvIndex : Integer); external 'dmath';
10179: {$ELSE}
10180: procedure PlotPoint(Xp, Yp, CurvIndex : Integer); external 'dmath';
10181: {$ENDIF}
10182: { Plots a point on the screen }
10183: procedure PlotCurve({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10184: X, Y : TVector;
10185: Lb, Ub, CurvIndex : Integer); external 'dmath';
10186: { Plots a curve }
10187: procedure PlotCurveWithErrorBars({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10188: X, Y, S : TVector;
10189: Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10190: { Plots a curve with error bars }
10191: procedure PlotFunc({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10192: Func : TFunc;
10193: Xmin, Xmax : Float;
10194: {$IFDEF DELPHI}Npt : Integer;{$ENDIF}
10195: CurvIndex : Integer); external 'dmath';
10196: { Plots a function }
10197: procedure WriteLegend({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10198: NCurv : Integer;
10199: ShowPoints, ShowLines : Boolean); external 'dmath';
10200: { Writes the legends for the plotted curves }
10201: procedure ConRec({$IFDEF DELPHI}Canvas : TCanvas;{$ENDIF}
10202: Nx, Ny, Nc : Integer;
10203: X, Y, Z : TVector;
10204: F : TMatrix); external 'dmath';
10205: { Contour plot }
10206: function Xpixel(X : Float):Integer; external 'dmath'; {Converts user abscissa X to screen coordinate }
10207: function Ypixel(Y : Float):Integer; external 'dmath'; {Converts user ordinate Y to screen coordinate }
10208: function Xuser(X : Integer):Float; external 'dmath'; {Converts screen coordinate X to user abscissa }
10209: function Yuser(Y : Integer):Float; external 'dmath'; {Converts screen coordinate Y to user ordinate }
10210: {$IFDEF DELPHI}
10211: procedure LeaveGraphics; external 'dmath';
10212: { Quits graphic mode }
10213: {$ENDIF}
10214: { -----
10215: LaTeX graphics
10216: ----- }
10217: function TeX_InitGraphics(FileName : String; PgWidth, PgHeight : Integer;
10218: Header : Boolean) : Boolean; external 'dmath';
10219: { Initializes the LaTeX file }
10220: procedure TeX_SetWindow(X1, X2, Y1, Y2 : Integer; GraphBorder : Boolean); external 'dmath';
10221: { Sets the graphic window }
10222: procedure TeX_LeaveGraphics(Footer : Boolean); external 'dmath'; { Close the LaTeX file }
10223: procedure TeX_SetOxScale(Scale : TScale; OxMin, OxMax, OxStep : Float); external 'dmath';
10224: { Sets the scale on the Ox axis }
10225: procedure TeX_SetOyScale(Scale : TScale; OyMin, OyMax, OyStep : Float); external 'dmath';
10226: { Sets the scale on the Oy axis }
10227: procedure TeX_SetGraphTitle(Title : String); external 'dmath'; { Sets the title for the graph }
10228: procedure TeX_SetOxTitle(Title : String); external 'dmath'; { Sets the title for the Ox axis }
10229: procedure TeX_SetOyTitle(Title : String); external 'dmath'; { Sets the title for the Oy axis }
10230: procedure TeX_PlotOxAxis; external 'dmath'; { Plots the horizontal axis }

```

```

10231: procedure TeX_PlotOyAxis; external 'dmath'; { Plots the vertical axis }
10232: procedure TeX_PlotGrid(Grid : TGrid); external 'dmath'; { Plots a grid on the graph }
10233: procedure TeX_WriteGraphTitle; external 'dmath'; Writes the title of the graph }
10234: procedure TeX_SetMaxCurv(NCurv : Byte); external 'dmath';
10235: { Sets the maximum number of curves and re-initializes their parameters }
10236: procedure TeX_SetPointParam(CurvIndex, Symbol, Size : Integer); external 'dmath';
10237: { Sets the point parameters for curve # CurvIndex }
10238: procedure TeX_SetLineParam(CurvIndex, Style : Integer;
10239:   Width : Float; Smooth : Boolean); external 'dmath';
10240: { Sets the line parameters for curve # CurvIndex }
10241: procedure TeX_SetCurvLegend(CurvIndex : Integer; Legend : String); external 'dmath';
10242: { Sets the legend for curve # CurvIndex }
10243: procedure TeX_SetCurvStep(CurvIndex, Step : Integer); external 'dmath';
10244: { Sets the step for curve # CurvIndex }
10245: procedure TeX_PlotCurve(X, Y : TVector; Lb, Ub, CurvIndex : Integer); external 'dmath';
10246: { Plots a curve }
10247: procedure TeX_PlotCurveWithErrorBars(X, Y, S : TVector;
10248:   Ns, Lb, Ub, CurvIndex : Integer); external 'dmath';
10249: { Plots a curve with error bars }
10250: procedure TeX_PlotFunc(Func : TFunc; X1, X2 : Float;
10251:   Npt : Integer; CurvIndex : Integer); external 'dmath';
10252: { Plots a function }
10253: procedure TeX_WriteLegend(NCurv : Integer; ShowPoints, ShowLines : Boolean); external 'dmath';
10254: { Writes the legends for the plotted curves }
10255: procedure TeX_ConRec(Nx, Ny, Nc : Integer; X, Y, Z : TVector; F : TMatrix); external 'dmath';
10256: { Contour plot }
10257: function Xcm(X : Float) : Float; external 'dmath'; { Converts user coordinate X to cm }
10258: function Ycm(Y : Float) : Float; external 'dmath'; { Converts user coordinate Y to cm }
10259:
10260: //*****unit uPSI_SynPdf;
10261: function RawUTF8ToPDFString( const Value : RawUTF8 ) : PDFString
10262: function _DateToPdfDate( ADate : TDateTime ) : TPDFDate
10263: function _PdfDateToDateTime( const AText : TPDFDate ) : TDateTime
10264: function PdfRect( Left, Top, Right, Bottom : Single ) : TPDFRect;
10265: function PdfRect1( const Box : TPDFBox ) : TPDFRect;
10266: function PdfBox( Left, Top, Width, Height : Single ) : TPDFBox
10267: //Function _GetCharCount( Text : PAnsiChar ) : integer
10268: //Procedure L2R( W : PWideChar; L : integer)
10269: function PdfCoord( MM : single ) : integer
10270: function CurrentPrinterPaperSize : TPDFPaperSize
10271: function CurrentPrinterRes : TPoint
10272: procedure GDICommentBookmark( MetaHandle : HDC; const aBookmarkName : RawUTF8 )
10273: procedure GDICommentOutline( MetaHandle : HDC; const aTitle : RawUTF8; aLevel : Integer )
10274: procedure GDICommentLink( MetaHandle:HDC; const aBookmarkName:RawUTF8; const aRect : TRect )
10275: const('Usp10','String').SetString( 'usp10.dll
10276: AddTypeS('TScriptState_enum', '( r0, r1, r2, r3, r4, fOverrideDirection, fInhibitSymSwap,
10277:   'fCharShape, fDigitSubstitute,fInhibitLigate,fDisplayZWG, fArabicNumContext, fGcpClusters )
10278: AddTypeS('TScriptState_set', 'set of TScriptState_enum
10279: //*****
10280:
10281: procedure SIRegister_PMrand(CL: TPSPascalCompiler); //ParkMiller
10282: begin
10283:   procedure PMrandomize( I : word)
10284:   function PMrandom : longint
10285:   function Rrand : extended
10286:   function Irand( N : word) : word
10287:   function Brand( P : extended) : boolean
10288:   function Nrand : extended
10289: end;
10290:
10291: procedure SIRegister_Spring_Cryptography_Utils(CL: TPSPascalCompiler);
10292: begin
10293:   function Endian( x : LongWord) : LongWord
10294:   function Endian64( x : Int64) : Int64
10295:   function spRol( x : LongWord; y : Byte) : LongWord
10296:   function spRor( x : LongWord; y : Byte) : LongWord
10297:   function Ror64( x : Int64; y : Byte) : Int64
10298: end;
10299:
10300: procedure SIRegister_MapReader(CL: TPSPascalCompiler);
10301: begin
10302:   procedure ClearModules
10303:   procedure ReadMapFile( FName : string)
10304:   function AddressInfo( Address : dword) : string
10305: end;
10306:
10307: procedure SIRegister_LibTar(CL: TPSPascalCompiler);
10308: begin
10309:   TTarPermission, '( tpReadByOwner, tpWriteByOwner, tpExecuteByOw'
10310:   + 'ner, tpReadByGroup, tpWriteByGroup, tpExecuteByGroup, tpReadByOther, tpWri'
10311:   + 'teByOther, tpExecuteByOther )
10312:   TTarPermissions, 'set of TTarPermission
10313:   TFileType, '( ftNormal, ftLink, ftSymbolicLink, ftCharacter, ft'
10314:   + 'Block, ftDirectory, ftFifo, ftContiguous, ftDumpDir, ftMultiVolume, ftVolumeHeader;
10315:   TTarMode, '( tmSetUid, tmSetGid, tmSaveText )
10316:   TTarModes, 'set of TTarMode
10317:   TTarDirRec, 'record Name : STRING; Size : INT64; DateTime : TDa'
10318:   + 'teTime; Permissions : TTarPermissions; FileType : TFileType; LinkName : ST'
10319:   + 'RING; UID : INTEGER; GID : INTEGER; UserName : STRING; GroupName : STRING;

```

```

10320:   +'ChecksumOK : BOOLEAN; Mode : TTarModes; Magic : STRING; MajorDevNo : INTE'
10321:   +'GER; MinorDevNo : INTEGER; FilePos : INT64; end
10322:   SIRegister_TTarArchive(CL);
10323:   SIRegister_TTarWriter(CL);
10324:   Function PermissionString( Permissions : TTarPermissions) : STRING
10325:   Function ConvertFilename( Filename : STRING) : STRING
10326:   Function FileTimeGMT( FileName : STRING) : TDateTime;
10327:   Function FileTimeGMT1( SearchRec : TSearchRec) : TDateTime;
10328:   Procedure ClearDirRec( var DirRec : TTarDirRec)
10329: end;
10330:
10331:
10332: //*****unit uPSI_TlHelp32;
10333: procedure SIRegister_TlHelp32(CL: TPSPascalCompiler);
10334: begin
10335:   Const('MAX_MODULE_NAME32','LongInt').SetInt( 255);
10336:   Function CreateToolhelp32Snapshot( dwFlags, th32ProcessID : DWORD) : THandle
10337:   Const('TH32CS_SNAPHEAPLIST','LongWord').SetUInt( $00000001);
10338:   Const('TH32CS_SNAPPROCESS','LongWord').SetUInt( $00000002);
10339:   Const('TH32CS_SNAPTHREAD','LongWord').SetUInt( $00000004);
10340:   Const('TH32CS_SNAPMODULE','LongWord').SetUInt( $00000008);
10341:   Const('TH32CS_INHERIT','LongWord').SetUInt( $80000000);
10342:   tagHEAPLIST32,'record dwSize:DWORD;th32ProcessID:DWORD;th32HeapID:DWORD;dwFlags:DWORD;end';
10343:   AddTypeS('HEAPLIST32','tagHEAPLIST32
10344:   AddTypeS('THeapList32','tagHEAPLIST32
10345:   Const('HF32_DEFAULT','LongInt').SetInt( 1);
10346:   Const('HF32_SHARED','LongInt').SetInt( 2);
10347:   Function Heap32ListFirst( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10348:   Function Heap32ListNext( hSnapshot : THandle; var lphl : THeapList32) : BOOL
10349:   AddTypeS('tagHEAPENTRY32','record dwSize : DWORD; hHandle : THandle; dwAd'
10350:   +'dress : DWORD; dwBlockSize : DWORD; dwFlags : DWORD; dwLockCount : DWORD; '
10351:   +'dwResvd : DWORD; th32ProcessID : DWORD; th32HeapID : DWORD; end
10352:   AddTypeS('HEAPENTRY32','tagHEAPENTRY32
10353:   AddTypeS('THeapEntry32','tagHEAPENTRY32
10354:   Const('LF32_FIXED','LongWord').SetUInt( $00000001);
10355:   Const('LF32_FREE','LongWord').SetUInt( $00000002);
10356:   Const('LF32_MOVEABLE','LongWord').SetUInt( $00000004);
10357:   Function Heap32First( var lphe : THeapEntry32; th32ProcessID, th32HeapID : DWORD) : BOOL
10358:   Function Heap32Next( var lphe : THeapEntry32) : BOOL
10359:   DWORD; var lpNumberOfBytesRead : DWORD) : BOOL
10360:   AddTypeS('tagTHREADENTRY32','record dwSize : DWORD; cntUsage : DWORD; th3'
10361:   +'2ThreadID : DWORD; th32OwnerProcessID : DWORD; tpBasePri : Longint; tpDelt'
10362:   +'aPri : Longint; dwFlags : DWORD; end
10363:   AddTypeS('THREADENTRY32','tagTHREADENTRY32
10364:   AddTypeS('TThreadEntry32','tagTHREADENTRY32
10365:   Function Thread32First( hSnapshot : THandle; var lpte : TThreadEntry32) : BOOL
10366:   Function Thread32Next( hSnapshot : THandle; var lpte : TThreadEntry32) : BOOL
10367: end;
10368:   Const('EW_RESTARTWINDOWS','LongWord').SetUInt( $0042);
10369:   Const('EW_REBOOTSYSTEM','LongWord').SetUInt( $0043);
10370:   Const('EW_EXITANDEXECAPP','LongWord').SetUInt( $0044);
10371:   Const('ENDSESSION_LOGOFF','LongWord').SetUInt( DWORD ( $80000000 ));
10372:   Const('EWX_LOGOFF','LongInt').SetInt( 0);
10373:   Const('EWX_SHUTDOWN','LongInt').SetInt( 1);
10374:   Const('EWX_REBOOT','LongInt').SetInt( 2);
10375:   Const('EWX_FORCE','LongInt').SetInt( 4);
10376:   Const('EWX_POWEROFF','LongInt').SetInt( 8);
10377:   Const('EWX_FORCEIFHUNG','LongWord').SetUInt( $10);
10378:   Function GET_APPCOMMAND_LPARAM( const lParam : LongInt) : Shortint
10379:   Function GET_DEVICE_LPARAM( const lParam : LongInt) : Word
10380:   Function GET_MOUSEORKEY_LPARAM( const lParam : LongInt) : Word
10381:   Function GET_FLAGS_LPARAM( const lParam : LongInt) : Word
10382:   Function GET_KEYSTATE_LPARAM( const lParam : LongInt) : Word
10383:   Function GetWindowWord( hWnd : HWND; nIndex : Integer) : Word
10384:   Function SetWindowWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10385:   Function GetWindowLong( hWnd : HWND; nIndex : Integer) : Longint
10386:   Function SetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
10387:   Function GetClassWord( hWnd : HWND; nIndex : Integer) : Word
10388:   Function SetClassWord( hWnd : HWND; nIndex : Integer; wNewWord : Word) : Word
10389:   Function GetClassLong( hWnd : HWND; nIndex : Integer) : DWORD
10390:   Function SetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : DWORD
10391:   Function GetDesktopWindow : HWND
10392:   Function GetParent( hWnd : HWND) : HWND
10393:   Function SetParent( hWndChild, hWndNewParent : HWND) : HWND
10394:   Function GetTopWindow( hWnd : HWND) : HWND
10395:   Function GetNextWindow( hWnd : HWND; uCmd : UINT) : HWND
10396:   Function GetWindow( hWnd : HWND; uCmd : UINT) : HWND
10397: //Delphi DFM
10398:   Function LoadDFMFile2Strings(const AFile:string; AStrings:TStrings; var WasText:boolean):integer
10399:   Function SaveStrings2DFMFile( AStrings : TStrings; const AFile : string) : integer
10400:   procedure GetHighlighters(AOwner: TComponent; AHighlighters: TStringList; AppendToList: boolean);
10401:   function GetHighlightersFilter(AHighlighters: TStringList): string;
10402:   function GetHighlighterFromFileExt(AHighlighters: TStringList;Extension: string):TSynCustomHighlighter;
10403:   Function ShowOwnedPopups( hWnd : HWND; fShow : BOOL) : BOOL
10404:   Function OpenIcon( hWnd : HWND) : BOOL
10405:   Function CloseWindow( hWnd : HWND) : BOOL
10406:   Function MoveWindow( hWnd : HWND; X, Y, nWidth, nHeight : Integer; bRepaint : BOOL) : BOOL
10407:   Function SetWindowPos(hWnd: HWND;hWndInsertAfter:HWND; X,Y,cx,cy : Integer; uFlags : UINT) : BOOL
10408:   Function IsWindowVisible( hWnd : HWND) : BOOL

```



```

10409: Function IsIconic( hWnd : HWND ) : BOOL
10410: Function AnyPopup : BOOL
10411: Function BringWindowToTop( hWnd : HWND ) : BOOL
10412: Function IsZoomed( hWnd : HWND ) : BOOL
10413: Function IsWindow( hWnd : HWND ) : BOOL
10414: Function IsMenu( hMenu : HMENU ) : BOOL
10415: Function IsChild( hWndParent, hWnd : HWND ) : BOOL
10416: Function DestroyWindow( hWnd : HWND ) : BOOL
10417: Function ShowWindow( hWnd : HWND; nCmdShow : Integer ) : BOOL
10418: Function AnimateWindow( hWnd : HWND; dwTime : DWORD; dwFlags : DWORD ) : BOOL
10419: Function ShowWindowAsync( hWnd : HWND; nCmdShow : Integer ) : BOOL
10420: Function FlashWindow( hWnd : HWND; bInvert : BOOL ) : BOOL
10421: Function IsWindowUnicode( hWnd : HWND ) : BOOL
10422: Function EnableWindow( hWnd : HWND; bEnable : BOOL ) : BOOL
10423: Function IsWindowEnabled( hWnd : HWND ) : BOOL
10424:
10425: procedure SIRegister_IDECmdLine(CL: TPSPascalCompiler);
10426: begin
10427:   const ('ShowSetupDialogOptLong', 'String').SetString( '--setup
10428:   PrimaryConfPathOptLong', 'String').SetString( '--primary-config-path=
10429:   PrimaryConfPathOptShort', 'String').SetString( '--pcp=
10430:   SecondaryConfPathOptLong', 'String').SetString( '--secondary-config-path=
10431:   SecondaryConfPathOptShort', 'String').SetString( '--scp=
10432:   NoSplashScreenOptLong', 'String').SetString( '--no-splash-screen
10433:   NoSplashScreenOptShort', 'String').SetString( '--nsc
10434:   StartedByStartLazarusOpt', 'String').SetString( '--started-by-startlazarus
10435:   SkipLastProjectOpt', 'String').SetString( '--skip-last-project
10436:   DebugLogOpt', 'String').SetString( '--debug-log=
10437:   DebugLogOptEnable', 'String').SetString( '--debug-enable=
10438:   LanguageOpt', 'String').SetString( '--language=
10439:   LazarusDirOpt', 'String').SetString( '--lazarusdir=
10440: Procedure ParseCommandLine(aCmdLineParams:TStrings;out IDEPid:Int;out ShowSplashScreen:boolean);
10441: Function GetCommandLineParameters( aCmdLineParams : TStrings; isStartLazarus:Boolean) : string
10442: Function ExtractPrimaryConfigPath( aCmdLineParams : TStrings) : string
10443: Function IsHelpRequested : Boolean
10444: Function IsVersionRequested : boolean
10445: Function GetLanguageSpecified : string
10446: Function ParamIsOption( ParamIndex : integer; const Option : string) : boolean
10447: Function ParamIsOptionPlusValue(ParamIndex:integer;const Option:string;out AValue:string):bool;
10448: Procedure ParseNoGuiCmdLineParams
10449: Function ExtractCmdLineFileNames : TStrings
10450: end;
10451:
10452:
10453: procedure SIRegister_LazFileUtils(CL: TPSPascalCompiler);
10454: begin
10455:   Function CompareFileNames( const Filename1, Filename2 : string) : integer
10456:   Function CompareFileNamesIgnoreCase( const Filename1, Filename2 : string) : integer
10457:   Function CompareFileExt( const Filename, Ext : string; CaseSensitive : boolean) : integer;
10458:   Function CompareFileExt1( const Filename, Ext : string) : integer;
10459:   Function CompareFilenameStarts( const Filename1, Filename2 : string) : integer
10460:   Function CompareFileNames(Filename1:PChar;Len1:integer; Filename2:PChar;Len2:integer):integer
10461:   Function CompareFileNamesP( Filename1, Filename2 : PChar; IgnoreCase : boolean) : integer
10462:   Function DirPathExists( DirectoryName : string) : boolean
10463:   Function DirectoryIsWritable( const DirectoryName : string) : boolean
10464:   Function ExtractFileNameOnly( const AFilename : string) : string
10465:   Function FilenameIsAbsolute( const TheFilename : string) : boolean
10466:   Function FilenameIsWinAbsolute( const TheFilename : string) : boolean
10467:   Function FilenameIsUnixAbsolute( const TheFilename : string) : boolean
10468:   Function ForceDirectory( DirectoryName : string) : boolean
10469:   Procedure CheckIfFileIsExecutable( const AFilename : string)
10470:   Procedure CheckIfFileIsSymlink( const AFilename : string)
10471:   Function FileIsText( const AFilename : string) : boolean
10472:   Function FileIsText2( const AFilename : string; out FileReadable : boolean) : boolean
10473:   Function FilenameIsTrimmed( const TheFilename : string) : boolean
10474:   Function FilenameIsTrimmed2( StartPos : PChar; NameLen : integer) : boolean
10475:   Function TrimFilename( const AFilename : string) : string
10476:   Function ResolveDots( const AFilename : string) : string
10477:   Procedure ForcePathDelims( var FileName : string)
10478:   Function GetForcedPathDelims( const FileName : string) : string
10479:   Function CleanAndExpandFilename( const Filename : string) : string
10480:   Function CleanAndExpandDirectory( const Filename : string) : string
10481:   Function TrimAndExpandFilename( const Filename : string; const BaseDir : string) : string
10482:   Function TrimAndExpandDirectory( const Filename : string; const BaseDir : string) : string
10483:   Function TryCreateRelativePath(const Dest,Source:string; UsePointDirectory:bool;
AlwaysRequireSharedBaseFolder : Boolean; out RelPath : string) : Boolean
10484:   Function CreateRelativePath( const Filename,BaseDirectory:string; UsePointDirectory:boolean;
AlwaysRequireSharedBaseFolder: Boolean) : string
10485:   Function FileIsInPath( const Filename, Path : string) : boolean
10486:   Function AppendPathDelim( const Path : string) : string
10487:   Function ChompPathDelim( const Path : string) : string
10488:   Function CreateAbsoluteSearchPath( const SearchPath, BaseDirectory : string) : string
10489:   Function CreateRelativeSearchPath( const SearchPath, BaseDirectory : string) : string
10490:   Function MinimizeSearchPath( const SearchPath : string) : string
10491:   Function FindPathInSearchPath(APath:PChar;APathLen:int;SearchPath:PChar;SearchPathLen:int):PChar;
10492:   (*Function FileExistsUTF8( const Filename : string) : boolean
10493:   Function FileAgeUTF8( const FileName : string) : Longint
10494:   Function DirectoryExistsUTF8( const Directory : string) : Boolean
10495:   Function ExpandFileNameUTF8( const FileName : string; BaseDir : string) : string

```

```

10496: Function FindFirstUTF8(const Path:string; Attr: Longint; out Rslt : TSearchRec) : Longint
10497: Function FindNextUTF8( var Rslt : TSearchRec) : Longint
10498: Procedure FindCloseUTF8( var F : TSearchRec)
10499: Function FileSetDateUTF8( const FileName : String; Age : Longint) : Longint
10500: Function FileGetAttrUTF8( const FileName : String) : Longint
10501: Function FileSetAttrUTF8( const FileName : String; Attr : longint) : Longint
10502: Function DeleteFileUTF8( const FileName : String) : Boolean
10503: Function RenameFileUTF8( const OldName, NewName : String) : Boolean
10504: Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean) : String
10505: Function FileIsReadOnlyUTF8( const FileName : String) : Boolean
10506: Function GetCurrentDirUTF8 : String
10507: Function SetCurrentDirUTF8( const NewDir : String) : Boolean
10508: Function CreateDirUTF8( const NewDir : String) : Boolean
10509: Function RemoveDirUTF8( const Dir : String) : Boolean
10510: Function ForceDirectoriesUTF8( const Dir : string) : Boolean
10511: Function FileOpenUTF8( const FileName : string; Mode : Integer) : THandle
10512: Function FileCreateUTF8( const FileName : string) : THandle;
10513: Function FileCreateUTF8( const FileName : string; Rights : Cardinal) : THandle;
10514: Function FileCreateUtf82( const FileName : String; ShareMode : Integer; Rights : Cardinal) : THandle;
10515: Function FileSizeUtf8( const Filename : string) : int64
10516: Function GetFileDescription( const AFilename : string) : string
10517: Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean) : string
10518: Function GetAppConfigFileUTF8( Global : Boolean; SubDir : boolean; CreateDir : boolean) : string
10519: Function GetTempFileNameUTF8( const Dir, Prefix : String) : String*)
10520: Function IsUNCPath( const Path : String) : Boolean
10521: Function ExtractUNCVolume( const Path : String) : String
10522: Function ExtractFileRoot( FileName : String) : String
10523: Function GetDarwinSystemFilename( Filename : string) : string
10524: Procedure SplitCmdLineParams( const Params : string; ParamList : TStrings; ReadBackslash : boolean)
10525: Function StrToCmdLineParam( const Param : string) : string
10526: Function MergeCmdLineParams( ParamList : TStrings) : string
10527: Procedure InvalidateFileStateCache( const Filename : string)
10528: Function FindAllFiles(const SearchPath:String;SearchMask:String;SearchSubDirs:Boolean):TStringList);
10529: Function FindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean) : TStringList
10530: Function ReadFileToString( const Filename : string) : string
10531: type
10532:   TCopyFileFlag = ( cffOverwriteFile,
10533:     cffCreateDestDirectory, cffPreserveTime );
10534:   TCopyFileFlags = set of TCopyFileFlag;*)
10535:   TCopyFileFlag', '(cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime)
10536:   TCopyFileFlags', 'set of TCopyFileFlag
10537:   Function CopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags) : Boolean
10538: end;
10539:
10540: procedure SIRegister_lazMasks(CL: TPSPascalCompiler);
10541: begin
10542:   TMaskCharType', '( mcChar, mcCharSet, mcAnyChar, mcAnyText )
10543:   SIRegister_TMask(CL);
10544:   SIRegister_TParseStringList(CL);
10545:   SIRegister_TMaskList(CL);
10546:   Function MatchesMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Boolean
10547:   Function MatchesWindowsMask( const FileName, Mask : String; const CaseSensitive : Boolean) : Bool;
10548:   Function MatchesMaskList(const FileName,Mask:String;Separator:Char;const CaseSensitive:Boolean):Bool;
10549:   Function MatchesWindowsMaskList(const FileName,Mask:String;Separat:Char;const CaseSensitive:Bool):Bool;
10550: end;
10551:
10552: procedure SIRegister_JvShellHook(CL: TPSPascalCompiler);
10553: begin
10554:   //PShellHookInfo', '^TShellHookInfo // will not work
10555:   TShellHookInfo', 'record hwnd : THandle; rc : TRect; end
10556:   SHELLHOOKINFO', 'TShellHookInfo
10557:   LPSHELLHOOKINFO', 'PShellHookInfo
10558:   TJvShellHookEvent', 'Procedure ( Sender : TObject; var Msg : TMessage)
10559:   SIRegister_TJvShellHook(CL);
10560:   Function InitJvShellHooks : Boolean
10561:   Procedure UnInitJvShellHooks
10562: end;
10563:
10564: procedure SIRegister_JvExControls(CL: TPSPascalCompiler);
10565: begin
10566:   TDlgCode', '( dcWantAllKeys, dcWantArrows, dcWantChars, dcButton'
10567:   +', dcHasSetSel, dcWantTab, dcNative )
10568:   TDlgCodes', 'set of TDlgCode
10569:   'dcWantMessage', '').SetString( dcWantAllKeys);
10570:   SIRegister_IJvExControl(CL);
10571:   SIRegister_IJvDenySubClassing(CL);
10572:   SIRegister_TStructPtrMessage(CL);
10573:   Procedure SetDotNetFrameColors( FocusedColor, UnfocusedColor : TColor)
10574:   Procedure DrawDotNetControl( Control : TWinControl; AColor : TColor; InControl : Boolean);
10575:   Procedure DrawDotNetControll( DC : HDC; R : TRect; AColor : TColor; UseFocusedColor : Boolean);
10576:   Procedure HandleDotNetHighlighting(Control:TWinControl;const Msg:TMessage;MouseOver:Boolean;Color:TColor);
10577:   Function CreateWMMMessage( Msg : Integer; WParam : Integer; LParam : Longint) : TMessage;
10578:   Function CreateWMMMessage1( Msg : Integer; WParam : Integer; LParam : TControl) : TMessage;
10579:   Function SmallPointToLong( const Pt : TSmallPoint) : Longint
10580:   Function ShiftStateToKeyData( Shift : TShiftState) : Longint
10581:   Function GetFocusedControl( AControl : TControl) : TWinControl
10582:   Function DlgcToDlgCodes( Value : Longint) : TDlgCodes
10583:   Function DlgCodesToDlgc( Value : TDlgCodes) : Longint
10584:   Procedure GetHintColor( var HintInfo : THintInfo; AControl : TControl; HintColor : TColor)

```

```

10585: Function DispatchIsDesignMsg( Control : TControl; var Msg : TMessage) : Boolean
10586:   SIRegister_TJvExControl(CL);
10587:   SIRegister_TJvExWinControl(CL);
10588:   SIRegister_TJvExCustomControl(CL);
10589:   SIRegister_TJvExGraphicControl(CL);
10590:   SIRegister_TJvExHintWindow(CL);
10591:   SIRegister_TJvExPubGraphicControl(CL);
10592: end;
10593:
10594: (*-----*)
10595: procedure SIRegister_EncdDecd(CL: TPSPascalCompiler);
10596: begin
10597:   Procedure EncodeStream( Input, Output : TStream)
10598:   Procedure DecodeStream( Input, Output : TStream)
10599:   Function EncodeString1( const Input : string) : string
10600:   Function DecodeString1( const Input : string) : string
10601: end;
10602:
10603: (*-----*)
10604: procedure SIRegister_SockAppReg(CL: TPSPascalCompiler);
10605: begin
10606:   SIRegister_TWebAppRegInfo(CL);
10607:   SIRegister_TWebAppRegList(CL);
10608:   Procedure GetRegisteredWebApps( AList : TWebAppRegList)
10609:   Procedure RegisterWebApp( const AFileName, AProgID : string)
10610:   Procedure UnregisterWebApp( const AProgID : string)
10611:   Function FindRegisteredWebApp( const AProgID : string) : string
10612:   Function CreateRegistry( InitializeNewFile : Boolean) : TCustomIniFile
10613:   'sUDPPort', 'String').SetString( 'UDPPort
10614: end;
10615:
10616: procedure SIRegister_PJEnvVars(CL: TPSPascalCompiler);
10617: begin
10618:   // TStringDynArray', 'array of string
10619:   Function GetEnvVarValue( const VarName : string) : string
10620:   Function SetEnvVarValue( const VarName, VarValue : string) : Integer
10621:   Function DeleteEnvVar( const VarName : string) : Integer
10622:   Function CreateEnvBlock(const NewEnv:TStrings;const IncludeCurrent:Bool;const Buffer:string;const
BufSize:Int):Int;
10623:   Function ExpandEnvVars( const Str : string) : string
10624:   Function GetAllEnvVars( const Vars : TStrings) : Integer
10625:   Procedure GetAllEnvVarNames( const Names : TStrings);
10626:   Function GetAllEnvVarNames1 : TStringDynArray;
10627:   Function EnvBlockSize : Integer
10628:   TPJEnvVarsEnum', 'Procedure ( const VarName : string; Data : TObject)
10629:   SIRegister_TPJEnvVarsEnumerator(CL);
10630:   SIRegister_TPJEnvVars(CL);
10631:   FindClass('TObject'),'EPJEnvVars
10632:   FindClass('TObject'),'EPJEnvVars
10633:   //Procedure Register
10634: end;
10635:
10636: (*-----*)
10637: procedure SIRegister_PJConsoleApp(CL: TPSPascalCompiler);
10638: begin
10639:   'cOneSecInMS','LongInt').SetInt( 1000);
10640:   // 'cDefTimeSlice','LongInt').SetInt( 50);
10641:   // 'cDefMaxExecTime','').SetString( cOneMinInMS);
10642:   'cAppErrorMask','LongInt').SetInt( 1 shl 29);
10643:   Function IsApplicationError( const ErrCode : LongWord) : Boolean
10644:   TPJConsoleAppPriority', '( cpDefault, cpHigh, cpNormal, cpIdle, cpRealTime )
10645:   TPJConsoleColors', 'record Foreground : TPJConsoleColor; Background:TPJConsoleColor; end;
10646:   Function MakeConsoleColors( const AForeground, ABackground : TPJConsoleColor):TPJConsoleColors;
10647:   Function MakeConsoleColors1( const AForeground, ABackground : TColor) : TPJConsoleColors;
10648:   Function MakeConsoleColors2( const AForeground, ABackground : TAlphaColor) : TPJConsoleColors;
10649:   Function MakeSize( const ACX, ACY : LongInt) : TSize
10650:   SIRegister_TPJCustomConsoleApp(CL);
10651:   SIRegister_TPJConsoleApp(CL);
10652: end;
10653:
10654: procedure SIRegister_ip_misc(CL: TPSPascalCompiler);
10655: begin
10656:   INVALID_IP_ADDRESS','LongWord').SetUInt( $ffffff);
10657:   t_encoding', '( uencode, base64, mime )
10658:   Function internet_date( date : TDateTime) : string
10659:   Function lookup_hostname( const hostname : string) : longint
10660:   Function my_hostname : string
10661:   Function my_ip_address : longint
10662:   Function ip2string( ip_address : longint) : string
10663:   Function resolve_hostname( ip : longint) : string
10664:   Function address_from( const s : string; count : integer) : string
10665:   Function encode_base64( data : TStream) : TStringList
10666:   Function decode_base64( source : TStringList) : TMemoryStream
10667:   Function posn( const s, t : string; count : integer) : integer
10668:   Function poscn( c : char; const s : string; n : integer) : integer
10669:   Function filename_of( const s : string) : string
10670:   //Function trim( const s : string) : string
10671:   //Procedure setlength( var s : string; l : byte)
10672:   Function TimeZoneBias : longint

```

```

10673: Function eight2seven_quoteprint( const s : string) : string
10674: Function eight2seven_german( const s : string) : string
10675: Function seven2eight_quoteprint( const s : string) : string end;
10676: type in_addr, 'record s_bytes : array[1..4] of byte; end;
10677: Function socketerror : cint
10678: Function fpsocket( domain : cint; xtype : cint; protocol : cint) : cint
10679: Function frecv( s : cint; buf : __pointer; len : size_t; flags : cint) : ssize_t
10680: Function fsend( s : cint; msg : __pointer; len : size_t; flags : cint) : ssize_t
10681: //Function fbind( s : cint; addrx : psockaddr; addrlen : tsocklen) : cint
10682: Function flisten( s : cint; backlog : cint) : cint
10683: //Function faccept( s : cint; addrx : psockaddr; addrlen : plongint) : cint
10684: //Function fconnect( s : cint; name : psockaddr; namelen : tsocklen) : cint
10685: //Function fpgetsockname( s : cint; name : psockaddr; namelen : psocklen) : cint
10686: Function NetAddrToStr( Entry : in_addr) : String
10687: Function HostAddrToStr( Entry : in_addr) : String
10688: Function StrToHostAddr( IP : String) : in_addr
10689: Function StrToNetAddr( IP : String) : in_addr
10690: SOL_SOCKET, 'LongWord').SetUInt( $ffff);
10691: cint8', 'shortint
10692: cuint8', 'byte
10693: cchar', 'cint8
10694: cschar', 'cint8
10695: cuchar', 'cuint8
10696: cint16', 'smallint
10697: cuint16', 'word
10698: cshort', 'cint16
10699: csshort', 'cint16
10700: cushort', 'cuint16
10701: cint32', 'longint
10702: cuint32', 'longword
10703: cint', 'cint32
10704: csint', 'cint32
10705: cuint', 'cuint32
10706: csigned', 'cint
10707: cunsigned', 'cuint
10708: cint64', 'int64
10709: clonglong', 'cint64
10710: cslonglong', 'cint64
10711: cbool', 'longbool
10712: cfloat', 'single
10713: cdouble', 'double
10714: clongdouble', 'extended
10715:
10716: procedure SIRegister_uLkJSON(CL: TPSPascalCompiler);
10717: begin
10718:   TlkJSONtypes', '(jsBase, jsNumber, jsString, jsBoolean, jsNull, jsList, jsObject )
10719:   SIRegister_TlkJSONdotnetclass(CL);
10720:   SIRegister_TlkJSONbase(CL);
10721:   SIRegister_TlkJSONnumber(CL);
10722:   SIRegister_TlkJSONstring(CL);
10723:   SIRegister_TlkJSONboolean(CL);
10724:   SIRegister_TlkJSONnull(CL);
10725:   TlkJSONFuncEnum', 'Procedure ( ElName : string; Elem : TlkJSONba
10726:   + 'se; data : TObjct; var Continue : Boolean)
10727:   SIRegister_TlkJSONcustomlist(CL);
10728:   SIRegister_TlkJSONlist(CL);
10729:   SIRegister_TlkJSONobjectmethod(CL);
10730:   TlkHashItem', 'record hash : cardinal; index : Integer; end
10731:   TlkHashFunction', 'Function ( const ws : WideString) : cardinal
10732:   SIRegister_TlkHashTable(CL);
10733:   SIRegister_TlkBalTree(CL);
10734:   SIRegister_TlkJSONobject(CL);
10735:   SIRegister_TlkJSON(CL);
10736:   SIRegister_TlkJSONstreamed(CL);
10737:   Function GenerateReadableText( vObj : TlkJSONbase; var vLevel : Integer): string
10738: end;
10739:
10740: procedure SIRegister_ZSysUtils(CL: TPSPascalCompiler);
10741: begin
10742:   TZListSortCompare', 'Function (Item1, Item2 : TObjct): Integer
10743:   SIRegister_TZSortedList(CL);
10744:   Function zFirstDelimiter( const Delimiters, Str : string) : Integer
10745:   Function zLastDelimiter( const Delimiters, Str : string) : Integer
10746:   //Function MemLCompUnicode( P1, P2 : PWideChar; Len : Integer) : Boolean
10747:   //Function MemLCompAnsi( P1, P2 : PAnsiChar; Len : Integer) : Boolean
10748:   Function zStartsWith( const Str, SubStr : WideString) : Boolean;
10749:   Function StartsWith1( const Str, SubStr : RawByteString) : Boolean;
10750:   Function EndsWith( const Str, SubStr : WideString) : Boolean;
10751:   Function EndsWith1( const Str, SubStr : RawByteString) : Boolean;
10752:   Function SQLStrToFloatDef( Str : RawByteString; Def : Extended) : Extended;
10753:   Function SQLStrToFloatDef1( Str : String; Def : Extended) : Extended;
10754:   Function SQLStrToFloat( const Str : AnsiString) : Extended
10755:   //Function BufferToStr( Buffer : PWideChar; Length : LongInt) : string;
10756:   //Function BufferToStr1( Buffer : PAnsiChar; Length : LongInt) : string;
10757:   Function BufferToBytes( Buffer : TObjct; Length : LongInt) : TByteDynArray
10758:   Function StrToBoolEx( Str : string) : Boolean
10759:   Function BoolToStrEx( Bool : Boolean) : String
10760:   Function IsIpAddr( const Str : string) : Boolean
10761:   Function zSplitString( const Str, Delimiters : string) : TStrings

```



```

10762: Procedure PutSplitString( List : TStrings; const Str, Delimiters : string)
10763: Procedure AppendSplitString( List : TStrings; const Str, Delimiters : string)
10764: Function ComposeString( List : TStrings; const Delimiter : string) : string
10765: Function FloatToSQLStr( Value : Extended) : string
10766: Procedure PutSplitStringEx( List : TStrings; const Str, Delimiter : string)
10767: Function SplitStringEx( const Str, Delimiter : string) : TStrings
10768: Procedure AppendSplitStringEx( List : TStrings; const Str, Delimiter : string)
10769: Function zBytesToStr( const Value : TByteDynArray) : AnsiString
10770: Function zStrToBytes( const Value : AnsiString) : TByteDynArray;
10771: Function StrToBytes1( const Value : UTF8String) : TByteDynArray;
10772: Function StrToBytes2( const Value : RawByteString) : TByteDynArray;
10773: Function StrToBytes3( const Value : WideString) : TByteDynArray;
10774: Function StrToBytes4( const Value : UnicodeString) : TByteDynArray;
10775: Function BytesToVar( const Value : TByteDynArray) : Variant
10776: Function VarToBytes( const Value : Variant) : TByteDynArray
10777: Function AnsiSQLDateToDateTime( const Value : string) : TDateTime
10778: Function TimestampStrToDateTime( const Value : string) : TDateTime
10779: Function DateTimeToAnsiSQLDate( Value : TDateTime; WithMMSec : Boolean) : string
10780: Function EncodeCString( const Value : string) : string
10781: Function DecodeCString( const Value : string) : string
10782: Function ZReplaceChar( const Source, Target : Char; const Str : string) : string
10783: Function MemPas( Buffer : PChar; Length : LongInt) : string
10784: Procedure DecodeSQLVersioning(const FullVersion:Integer;out MajorVersion:Integer;out MinorVersion:Integer;out
SubVersion:Integer);
10785: Function EncodeSQLVersioning(const MajorVersion:Integer;const MinorVersion:Integer;const
SubVersion:Integer):Integer;
10786: Function FormatSQLVersion( const SQLVersion : Integer) : String
10787: Function ZStrToFloat( Value : AnsiChar) : Extended;
10788: Function ZStrToFloat1( Value : AnsiString) : Extended;
10789: Procedure ZSetString( const Src : AnsiChar; var Dest : AnsiString);
10790: Procedure ZSetString1( const Src : AnsiChar; const Len : Cardinal; var Dest : AnsiString);
10791: Procedure ZSetString2( const Src : AnsiChar; var Dest : UTF8String);
10792: Procedure ZSetString3( const Src : AnsiChar; const Len : Cardinal; var Dest : UTF8String);
10793: Procedure ZSetString4( const Src : AnsiChar; const Len : Cardinal; var Dest : WideString);
10794: Procedure ZSetString5( const Src : AnsiChar; var Dest : RawByteString);
10795: Procedure ZSetString6( const Src : AnsiChar; const Len : Cardinal; var Dest : RawByteString);
10796: end;
10797:
10798: unit uPSI_ZEncoding;
10799: Function StringToAnsiEx( const s : String; const FromCP, ToCP : Word) : RawByteString
10800: Function AnsiToStringEx( const s : RawByteString; const FromCP, ToCP : Word) : String
10801: Function ZRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10802: Function ZUnicodeToRaw( const US : WideString; CP : Word) : RawByteString
10803: Function ZConvertAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10804: Function ZConvertRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10805: Function ZConvertAnsiToUTF8( const Src : AnsiString) : UTF8String
10806: Function ZConvertUTF8ToAnsi( const Src : UTF8String) : AnsiString
10807: Function ZConvertRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10808: Function ZConvertUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10809: Function ZConvertRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10810: Function ZConvertStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10811: Function ZConvertStringToRawWithAutoEncode(const Src:String;const StringCP,RawCP:Word):RawByteString;
10812: Function ZConvertUTF8ToString( const Src : UTF8String; const StringCP : Word) : String
10813: Function ZConvertStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10814: Function ZConvertStringToUTF8WithAutoEncode( const Src : String; const StringCP : Word) : UTF8String
10815: Function ZConvertStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10816: Function ZConvertStringToAnsiWithAutoEncode( const Src : String; const StringCP : Word) : AnsiString
10817: Function ZConvertAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10818: Function ZConvertUnicodeToString( const Src : WideString; const StringCP : Word) : String
10819: Function ZConvertUnicodeToString_CPUTF8( const Src : WideString; const StringCP : Word) : String
10820: Function ZConvertStringToUnicode( const Src : String; const StringCP : Word) : WideString
10821: Function ZConvertString_CPUTF8ToUnicode( const Src : String; const StringCP : Word) : WideString
10822: Function ZConvertStringToUnicodeWithAutoEncode( const Src: String; const StringCP:Word):WideString
10823: Function ZMoveAnsiToRaw( const Src : AnsiString; const RawCP : Word) : RawByteString
10824: Function ZMoveRawToAnsi( const Src : RawByteString; const RawCP : Word) : AnsiString
10825: Function ZMoveAnsiToUTF8( const Src : AnsiString) : UTF8String
10826: Function ZMoveUTF8ToAnsi( const Src : UTF8String) : AnsiString
10827: Function ZMoveRawToUTF8( const Src : RawByteString; const CP : Word) : UTF8String
10828: Function ZMoveUTF8ToRaw( const Src : UTF8String; const CP : Word) : RawByteString
10829: Function ZMoveStringToAnsi( const Src : String; const StringCP : Word) : AnsiString
10830: Function ZMoveAnsiToString( const Src : AnsiString; const StringCP : Word) : String
10831: Function ZMoveRawToString( const Src : RawByteString; const RawCP, StringCP : Word) : String
10832: Function ZMoveStringToRaw( const Src : String; const StringCP, RawCP : Word) : RawByteString
10833: Function ZMoveUTF8ToString( const Src : UTF8String; StringCP : Word) : String
10834: Function ZMoveStringToUTF8( const Src : String; const StringCP : Word) : UTF8String
10835: Function ZUnknownRawToUnicode( const S : RawByteString; const CP : Word) : WideString
10836: Function ZUnknownRawToUnicodeWithAutoEncode( const S : RawByteString; const CP : Word) : WideString
10837: Function ZUnicodeToUnknownRaw( const US : WideString; CP : Word) : RawByteString
10838: Function ZDefaultSystemCodePage : Word
10839: Function ZCompatibleCodePages( const CP1, CP2 : Word) : Boolean
10840:
10841:
10842: procedure SIRegister_BoldComUtils(CL: TPSPascalCompiler);
10843: begin
10844:   'RPC_C_AUTHN_LEVEL_DEFAULT', 'LongInt').SetInt( 0);
10845:   ('RPC_C_AUTHN_LEVEL_NONE', 'LongInt').SetInt( 1);
10846:   ('RPC_C_AUTHN_LEVEL_CONNECT', 'LongInt').SetInt( 2);
10847:   ('RPC_C_AUTHN_LEVEL_CALL', 'LongInt').SetInt( 3);
10848:   ('RPC_C_AUTHN_LEVEL_PKT', 'LongInt').SetInt( 4);

```

```

10849: ('RPC_C_AUTHN_LEVEL_PKT_INTEGRITY','LongInt').SetInt( 5);
10850: ('RPC_C_AUTHN_LEVEL_PKT_PRIVACY','LongInt').SetInt( 6);
10851: (('alDefault','1').SetString( RPC_C_AUTHN_LEVEL_DEFAULT);
10852: ('alNone','2').SetString( RPC_C_AUTHN_LEVEL_NONE);
10853: ('alConnect','3').SetString( RPC_C_AUTHN_LEVEL_CONNECT);
10854: ('alCall','4').SetString( RPC_C_AUTHN_LEVEL_CALL);
10855: ('alPacket','5').SetString( RPC_C_AUTHN_LEVEL_PKT);
10856: ('alPacketIntegrity','6').SetString( RPC_C_AUTHN_LEVEL_PKT_INTEGRITY);
10857: ('alPacketPrivacy','7').SetString( RPC_C_AUTHN_LEVEL_PKT_PRIVACY);}
10858: ('RPC_C_IMP_LEVEL_DEFAULT','LongInt').SetInt( 0);
10859: ('RPC_C_IMP_LEVEL_ANONYMOUS','LongInt').SetInt( 1);
10860: ('RPC_C_IMP_LEVEL_IDENTIFY','LongInt').SetInt( 2);
10861: ('RPC_C_IMP_LEVEL_IMPERSONATE','LongInt').SetInt( 3);
10862: ('RPC_C_IMP_LEVEL_DELEGATE','LongInt').SetInt( 4);
10863: (('ilDefault','0').SetString( RPC_C_IMP_LEVEL_DEFAULT);
10864: ('ilAnonymous','1').SetString( RPC_C_IMP_LEVEL_ANONYMOUS);
10865: ('ilIdentify','2').SetString( RPC_C_IMP_LEVEL_IDENTIFY);
10866: ('ilImpersonate','3').SetString( RPC_C_IMP_LEVEL_IMPERSONATE);
10867: ('ilDelegate','4').SetString( RPC_C_IMP_LEVEL_DELEGATE);}
10868: ('EOAC_NONE','LongWord').SetUInt( $0);
10869: ('EOAC_DEFAULT','LongWord').SetUInt( $800);
10870: ('EOAC_MUTUAL_AUTH','LongWord').SetUInt( $1);
10871: ('EOAC_STATIC_CLOAKING','LongWord').SetUInt( $20);
10872: ('EOAC_DYNAMIC_CLOAKING','LongWord').SetUInt( $40);
10873: ('EOAC_ANY_AUTHORITY','LongWord').SetUInt( $80);
10874: ('RPC_C_AUTHN_WINNT','LongInt').SetInt( 10);
10875: ('RPC_C_AUTHN_NONE','LongInt').SetInt( 0);
10876: ('RPC_C_AUTHN_NAME','LongInt').SetInt( 1);
10877: ('RPC_C_AUTHN_DCE','LongInt').SetInt( 2);
10878: FindClass('TOBJECT'),'EBoldCom
10879: Function BoldVariantIsType( V : OleVariant; TypeCode : Integer) : Boolean
10880: Function BoldMemoryToVariant( const Buffer, BufSize : Integer) : OleVariant
10881: Function BoldStreamToVariant( Stream : TStream) : OleVariant
10882: Function BoldStringsToVariant( Strings : TStrings) : OleVariant
10883: Function BoldVariantToMemory( V : OleVariant; var Buffer, BufSize : Integer) : Integer
10884: Function BoldVariantToStream( V : OleVariant; Stream : TStream) : Integer
10885: Function BoldVariantArrayOfArraysOfStringToStrings( V : OleVariant; Strings : TStrings) : Integer
10886: Function BoldVariantIsNamedValues( V : OleVariant) : Boolean
10887: Function BoldCreateNamedValues(const Names:array of string;const Values:array of OleVariant):OleVariant;
10888: Function BoldGetNamedValue( Data : OleVariant; const Name : string) : OleVariant
10889: Procedure BoldSetNamedValue( Data : OleVariant; const Name : string; Value : OleVariant)
10890: Function BoldCreateGUID : TGUID
10891: Function BoldCreateComObject( const ClsId, Iid : TGUID; out Obj : variant; out Res : HRESULT) : Boolean
10892: Function BoldCreateRemoteComObject(const HostName:string;const ClsId,Iid:TGUID;out Obj:variant;out
Res:HRESULT):Bool;
10893: Procedure BoldInitializeComSecurity( AuthenticationLevel, ImpersonationLevel : longint)
10894: Procedure BoldSetSecurityForInterface(AuthenticationLevel,ImpersonationLevel:longint;Unk:IUnknown);
10895: end;
10896:
10897: (*-----*)
10898: procedure SIRegister_BoldIsoDateTime(CL: TPSPascalCompiler);
10899: begin
10900:   Function ParseISODate( s : string) : TDateTime
10901:   Function ParseISODateTime( s : string) : TDateTime
10902:   Function ParseISOTime( str : string) : TDateTime
10903: end;
10904:
10905: (*-----*)
10906: procedure SIRegister_BoldGUIDUtils(CL: TPSPascalCompiler);
10907: begin
10908:   Function BoldCreateGUIDAsString( StripBrackets : Boolean) : string
10909:   Function BoldCreateGUIDWithBracketsAsString : string
10910: end;
10911:
10912: procedure SIRegister_BoldFileHandler(CL: TPSPascalCompiler);
10913: begin
10914:   FindClass('TOBJECT'),'TBoldFileHandler
10915:   FindClass('TOBJECT'),'TBoldDiskFileHandler
10916:   //TBoldFileHandlerClass', 'class of TBoldFileHandler
10917:   TBoldInitializeFileContents', 'Procedure ( StringList : TStringList)
10918:   SIRegister_TBoldFileHandler(CL);
10919:   SIRegister_TBoldDiskFileHandler(CL);
10920:   Procedure BoldCloseAllFileHandlers
10921:   Procedure BoldRemoveUnchangedFilesFromEditor
10922:   Function BoldFileHandlerList : TBoldObjectArray
10923:   Function BoldFileHandlerForFile(path,FileName:String; ModuleType:TBoldModuleType;ShowInEditor:Bool;
OnInitializeFileContents : TBoldInitializeFileContents) : TBoldFileHandler
10924: end;
10925:
10926: procedure SIRegister_BoldWinINet(CL: TPSPascalCompiler);
10927: begin
10928:   PCharArr, 'array of PChar
10929:   Function BoldInternetOpen(Agent:String;
AccessType:integer;Proxy:String;ProxyByPass:String;Flags:integer):ptr);
10930:   Function BoldInternetOpenUrl(iNet:Pointer;URL: string; Headers:String;Flags,Context:cardinal):Pointer
10931:   Function BoldInternetReadFile(hFile:Pointer;Buff:Ptr;NumberOfBytesToRead:Card;var
NumberOfBytesRead:Card):LongBool;
10932:   Function BoldInternetCloseHandle( HINet : Pointer) : LongBool
10933:   Function BoldHttpRequestInfo( hRequest : Pointer; InfoLevel : Cardinal; Buffer : Pointer; BufferLength :
Cardinal; Reserved : Cardinal) : LongBool

```

```

10934: Function BoldInternetQueryDataAvailable( hFile : Pointer; var NumberOfBytesAvailable : Cardinal; flags :
Cardinal; Context : Cardinal) : LongBool
10935: Function BoldHttpOpenRequest(hConnect: Pointer; Verb, ObjectName, Version, Referrer : String; AcceptTypes
: PCharArr; Flags, Context : Cardinal) : Pointer
10936: Function BoldHttpSendRequest(hRequest:Pointer;Headers:string; Optional:Pointer;OptionalLength:Cardinal):
LongBool
10937: Function BoldInternetErrorDlg(hWnd:HWND;hRequest:HINTERNET;dwError,dwFlags:DWORD; var lppvData:Pointer):
DWORD
10938: Function BoldInternetAttemptConnect( dwReserved : DWORD) : DWORD
10939: Function BoldInternetConnect( hInet : HINTERNET; ServerName : string; nServerPort : INTERNET_PORT;
Username : string; Password : string; dwService : DWORD; dwFlags : DWORD; dwContext : DWORD) : HINTERNET
10940: Function BoldInternetCrackUrl(Url:PChar;UrlLength,dwFlags:DWORD; var lpUrlComponents:TURLComponents):BOOL;
10941: end;
10942:
10943: procedure SIRegister_BoldQueryUserDlg(CL: TPSPascalCompiler);
10944: begin
10945:   TBoldQueryResult', '( qrYesAll, qrYes, qrNo, qrNoAll )
10946:   SIRegister_TfrmBoldQueryUser(CL);
10947:   Function QueryUser( const Title, Query : string) : TBoldQueryResult
10948: end;
10949:
10950: (*-----*)
10951: procedure SIRegister_BoldQueue(CL: TPSPascalCompiler);
10952: begin
10953:   (('befIsInDisplayList','').SetString( BoldElementFlag0);
10954:   (('befStronglyDependedOfPrioritized','').SetString( BoldElementFlag1);
10955:   (('befFollowerSelected','').SetString( BoldElementFlag2);
10956:   FindClass('TOBJECT'),'TBoldQueue
10957:   FindClass('TOBJECT'),'TBoldQueueable
10958:   TBoldQueueDisplayMode', '( dmDisplayOne, dmDisplayAll )
10959:   SIRegister_TBoldQueueable(CL);
10960:   SIRegister_TBoldQueue(CL);
10961:   Function BoldQueueFinalized : Boolean
10962:   Function BoldInstalledQueue : TBoldQueue
10963: end;
10964:
10965: procedure SIRegister_Barcode(CL: TPSPascalCompiler);
10966: begin
10967:   const mmPerInch,'Extended').setExtended( 25.4);
10968:   TBarcodeType', '( bcCode_2_5_interleaved, bcCode_2_5_industrial, '
10969:   + ' bcCode_2_5_matrix, bcCode39, bcCode39Extended, bcCode128A, bcCode128B, bc'
10970:   + 'Code128C, bcCode93, bcCode93Extended, bcCodeMSI, bcCodePostNet, bcCodeCoda'
10971:   + 'bar, bcCodeEAN8, bcCodeEAN13, bcCodeUPC_A, bcCodeUPC_E0, bcCodeUPC_E1, bcC'
10972:   + 'odeUPC_Supp2, bcCodeUPC_Supp5, bcCodeEAN128A, bcCodeEAN128B, bcCodeEAN128C
10973:   TBarLineType', '( white, black, black_half )
10974:   TBarcodeOption', '( bcoNone, bcoCode, bcoTyp, bcoBoth )
10975:   TShowTextPosition', '( stpTopLeft, stpTopRight, stpTopCenter, st'
10976:   + 'pBottomLeft, stpBottomRight, stpBottomCenter )
10977:   TChecksumMethod', '( csmNone, csmModulo10 )
10978:   SIRegister_TASBarcode(CL);
10979:   Function CheckSumModulo10( const data : string) : string
10980:   Function ConvertMmToPixelsX( const Value : Double) : Integer
10981:   Function ConvertMmToPixelsY( const Value : Double) : Integer
10982:   Function ConvertInchToPixelsX( const Value : Double) : Integer
10983:   Function ConvertInchToPixelsY( const Value : Double) : Integer
10984: end;
10985:
10986: procedure SIRegister_Geometry(CL: TPSPascalCompiler); //OpenGL
10987: begin
10988:   THomogeneousByteVector', 'array[0..3] of Byte
10989:   THomogeneousWordVector', 'array[0..3] of Word
10990:   THomogeneousIntVector', 'array[0..3] of Integer
10991:   THomogeneousFltVector', 'array[0..3] of single
10992:   THomogeneousDblVector', 'array[0..3] of double
10993:   THomogeneousExtVector', 'array[0..3] of extended
10994:   TAffineByteVector', 'array[0..2] of Byte
10995:   TAffineWordVector', 'array[0..2] of Word
10996:   TAffineIntVector', 'array[0..2] of Integer
10997:   TAffineFltVector', 'array[0..2] of single
10998:   TAffineDblVector', 'array[0..2] of double
10999:   TAffineExtVector', 'array[0..2] of extended
11000:   THomogeneousByteMatrix', 'array[0..3] of THomogeneousByteVector
11001:   THomogeneousWordMatrix', 'array[0..3] of THomogeneousWordVector
11002:   THomogeneousIntMatrix', 'array[0..3] of THomogeneousIntVector
11003:   THomogeneousFltMatrix', 'array[0..3] of THomogeneousFltVector
11004:   THomogeneousDblMatrix', 'array[0..3] of THomogeneousDblVector
11005:   THomogeneousExtMatrix', 'array[0..3] of THomogeneousExtVector
11006:   TAffineByteMatrix', 'array[0..2] of TAffineByteVector
11007:   TAffineWordMatrix', 'array[0..2] of TAffineWordVector
11008:   TAffineIntMatrix', 'array[0..2] of TAffineIntVector
11009:   TAffineFltMatrix', 'array[0..3] of TAffineFltVector
11010:   TAffineDblMatrix', 'array[0..3] of TAffineDblVector
11011:   TAffineExtMatrix', 'array[0..3] of TAffineExtVector
11012:   TMatrix4b', 'THomogeneousByteMatrix
11013:   TMatrix4w', 'THomogeneousWordMatrix
11014:   TMatrix4i', 'THomogeneousIntMatrix
11015:   TMatrix4f', 'THomogeneousFltMatrix
11016:   TMatrix4d', 'THomogeneousDblMatrix
11017:   TMatrix4e', 'THomogeneousExtMatrix

```

```

11018:   TMatrix3b', 'TAffineByteMatrix
11019:   TMatrix3w', 'TAffineWordMatrix
11020:   TMatrix3i', 'TAffineIntMatrix
11021:   TMatrix3f', 'TAffineFltMatrix
11022:   TMatrix3d', 'TAffineDblMatrix
11023:   TMatrix3e', 'TAffineExtMatrix
11024:   ///'PMatrix', '^TMatrix // will not work
11025:   TMatrixGL', 'THomogeneousFltMatrix
11026:   THomogeneousMatrix', 'THomogeneousFltMatrix
11027:   TAffineMatrix', 'TAffineFltMatrix
11028:   TQuaternion', 'record Vector : TVector4f; end
11029:   TRectangle', 'record Left : integer; Top : integer; Width : inte'
11030:   +'ger; Height : Integer; end
11031:   TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
11032:   + 'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
11033:   + ', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW )
11034:   'EPSILON', 'Extended').setExtended( 1E-100);
11035:   'EPSILON2', 'Extended').setExtended( 1E-50);
11036:   Function VectorAddGL( V1, V2 : TVectorGL) : TVectorGL
11037:   Function VectorAffineAdd( V1, V2 : TAffineVector) : TAffineVector
11038:   Function VectorAffineCombine(V1,V2:TAffineVector; F1, F2 : Single) : TAffineVector
11039:   Function VectorAffineDotProduct( V1, V2 : TAffineVector) : Single
11040:   Function VectorAffineLerp( V1, V2 : TAffineVector; t : Single) : TAffineVector
11041:   Function VectorAffineSubtract( V1, V2 : TAffineVector) : TAffineVector
11042:   Function VectorAngle( V1, V2 : TAffineVector) : Single
11043:   Function VectorCombine( V1, V2 : TVectorGL; F1, F2 : Single) : TVectorGL
11044:   Function VectorCrossProduct( V1, V2 : TAffineVector) : TAffineVector
11045:   Function VectorDotProduct( V1, V2 : TVectorGL) : Single
11046:   Function VectorLength( V : array of Single) : Single
11047:   Function VectorLerp( V1, V2 : TVectorGL; t : Single) : TVectorGL
11048:   Procedure VectorNegate( V : array of Single)
11049:   Function VectorNorm( V : array of Single) : Single
11050:   Function VectorNormalize( V : array of Single) : Single
11051:   Function VectorPerpendicular( V, N : TAffineVector) : TAffineVector
11052:   Function VectorReflect( V, N : TAffineVector) : TAffineVector
11053:   Procedure VectorRotate( var Vector : TVector4f; Axis : TVector3f; Angle : Single)
11054:   Procedure VectorScale( V : array of Single; Factor : Single)
11055:   Function VectorSubtractGL( V1, V2 : TVectorGL) : TVectorGL
11056:   Function CreateRotationMatrixX( Sine, Cosine : Single) : TMatrixGL
11057:   Function CreateRotationMatrixY( Sine, Cosine : Single) : TMatrixGL
11058:   Function CreateRotationMatrixZ( Sine, Cosine : Single) : TMatrixGL
11059:   Function CreateScaleMatrix( V : TAffineVector) : TMatrixGL
11060:   Function CreateTranslationMatrix( V : TVectorGL) : TMatrixGL
11061:   Procedure MatrixAdjoint( var M : TMatrixGL)
11062:   Function MatrixAffineDeterminant( M : TAffineMatrix) : Single
11063:   Procedure MatrixAffineTranspose( var M : TAffineMatrix)
11064:   Function MatrixDeterminant( M : TMatrixGL) : Single
11065:   Procedure MatrixInvert( var M : TMatrixGL)
11066:   Function MatrixMultiply( M1, M2 : TMatrixGL) : TMatrixGL
11067:   Procedure MatrixScale( var M : TMatrixGL; Factor : Single)
11068:   Procedure MatrixTranspose( var M : TMatrixGL)
11069:   Function QuaternionConjugate( Q : TQuaternion) : TQuaternion
11070:   Function QuaternionFromPoints( V1, V2 : TAffineVector) : TQuaternion
11071:   Function QuaternionMultiply( qL, qR : TQuaternion) : TQuaternion
11072:   Function QuaternionSlerp( QStart,QEnd:TQuaternion; Spin:Integer; t:Single):TQuaternion
11073:   Function QuaternionToMatrix( Q : TQuaternion) : TMatrixGL
11074:   Procedure QuaternionToPoints( Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
11075:   Function ConvertRotation( Angles : TAffineVector) : TVectorGL
11076:   Function CreateRotationMatrix( Axis : TVector3f; Angle : Single) : TMatrixGL
11077:   //Function MatrixDecompose( M : TMatrixGL; var Tran : TTransformations) : Boolean
11078:   Function VectorAffineTransform( V : TAffineVector; M : TAffineMatrix) : TAffineVector
11079:   Function VectorTransform( V : TVector4f; M : TMatrixGL) : TVector4f;
11080:   Function VectorTransforml( V : TVector3f; M : TMatrixGL) : TVector3f;
11081:   Function MakeAffineDblVector( V : array of Double) : TAffineDblVector
11082:   Function MakeDblVector( V : array of Double) : THomogeneousDblVector
11083:   Function MakeAffineVector( V : array of Single) : TAffineVector
11084:   Function MakeQuaternion( Imag : array of Single; Real : Single) : TQuaternion
11085:   Function MakeVector( V : array of Single) : TVectorGL
11086:   Function PointInPolygonGL( xp, yp : array of Single; x, y : Single) : Boolean
11087:   Function VectorAffineDblToFlt( V : TAffineDblVector) : TAffineVector
11088:   Function VectorDblToFlt( V : THomogeneousDblVector) : THomogeneousVector
11089:   Function VectorAffineFltToDbl( V : TAffineVector) : TAffineDblVector
11090:   Function VectorFltToDbl( V : TVectorGL) : THomogeneousDblVector
11091:   Function ArcCosGL( X : Extended) : Extended
11092:   Function ArcSinGL( X : Extended) : Extended
11093:   Function ArcTan2GL( Y, X : Extended) : Extended
11094:   Function CoTanGL( X : Extended) : Extended
11095:   Function DegToRadGL( Degrees : Extended) : Extended
11096:   Function RadToDegGL( Radians : Extended) : Extended
11097:   Procedure SinCosGL( Theta : Extended; var Sin, Cos : Extended)
11098:   Function TanGL( X : Extended) : Extended
11099:   Function Turn( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11100:   Function Turnl( Matrix : TMatrixGL; MasterUp : TAffineVector; Angle: Single): TMatrixGL;
11101:   Function Pitch( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11102:   Function Pitchl( Matrix : TMatrixGL; MasterRight:TAffineVector;Angle:Single):TMatrixGL;
11103:   Function Roll( Matrix : TMatrixGL; Angle : Single) : TMatrixGL;
11104:   Function Rolll( Matrix:TMatrixGL; MasterDirection:TAffineVector;Angle:Single):TMatrixGL;
11105: end;
11106:

```



```

11107:
11108: procedure SIRegister_JclRegistry(CL: TPSPascalCompiler);
11109: begin
11110:   Function RegCreateKey( const RootKey : HKEY; const Key, Value : string) : Longint
11111:   Function RegDeleteEntry( const RootKey : HKEY; const Key, Name : string) : Boolean
11112:   Function RegDeleteKeyTree( const RootKey : HKEY; const Key : string) : Boolean
11113:   Function RegReadBool( const RootKey : HKEY; const Key, Name : string) : Boolean
11114:   Function RegReadBoolDef( const RootKey : HKEY; const Key, Name : string; Def : Boolean) : Boolean
11115:   Function RegReadInteger( const RootKey : HKEY; const Key, Name : string) : Integer
11116:   Function RegReadIntegerDef( const RootKey : HKEY; const Key, Name : string; Def : Integer) : Integer
11117:   Function RegReadString( const RootKey : HKEY; const Key, Name : string) : string
11118:   Function RegReadStringDef( const RootKey : HKEY; const Key, Name, Def : string) : string
11119:   Function RegReadDWORD( const RootKey : HKEY; const Key, Name : string) : Int64
11120:   Function RegReadDWORDDef( const RootKey : HKEY; const Key, Name : string; Def : Int64) : Int64
11121:   Procedure RegWriteBool( const RootKey : HKEY; const Key, Name : string; Value : Boolean)
11122:   Procedure RegWriteInteger( const RootKey : HKEY; const Key, Name : string; Value : Integer)
11123:   Procedure RegWriteString( const RootKey : HKEY; const Key, Name, Value : string)
11124:   Procedure RegWriteDWORD( const RootKey : HKEY; const Key, Name : string; Value : Int64)
11125:   Function RegGetValueNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11126:   Function RegGetKeyNames( const RootKey : HKEY; const Key : string; const List : TStrings) : Boolean
11127:   Function RegHasSubKeys( const RootKey : HKEY; const Key : string) : Boolean
11128:   Function RegKeyExists( const RootKey : HKEY; const Key : string) : Boolean
11129:   AddTypeS('TExecKind', '( ekMachineRun, ekMachineRunOnce, ekUserRun, ekUser'
11130:     + 'RunOnce, ekServiceRun, ekServiceRunOnce )
11131:   AddClassN(FindClass('TOBJECT'), 'EJclRegistryError
11132:   Function UnregisterAutoExec( ExecKind : TExecKind; const Name : string) : Boolean
11133:   Function RegisterAutoExec( ExecKind : TExecKind; const Name, Cmdline : string) : Boolean
11134:   Function RegSaveList(const RootKey:HKEY; const Key:string; const ListName:string;const
Items:TStrings):Bool;
11135:   Function RegLoadList(const RootKey:HKEY; const Key:string; const ListName:string;const
SaveTo:TStrings):Bool;
11136:   Function RegDelList( const RootKey:HKEY; const Key:string; const ListName:string): Boolean
11137: end;
11138:
11139: procedure SIRegister_JclCOM(CL: TPSPascalCompiler);
11140: begin
11141:   CLSID_StdComponentCategoriesMgr, 'TGUID').SetString( '{0002E005-0000-0000-C000-000000000046}'
11142:   CATID_SafeForInitializing, 'TGUID').SetString( '{7DD95802-9882-11CF-9FA9-00AA006C42C4}'
11143:   CATID_SafeForScripting, 'TGUID').SetString( '{7DD95801-9882-11CF-9FA9-00AA006C42C4}'
11144:   icMAX_CATEGORY_DESC_LEN, 'LongInt').SetInt( 128);
11145:   FindClass('TOBJECT'), 'EInvalidParam
11146:   Function IsDCOMInstalled : Boolean
11147:   Function IsDCOMEnabled : Boolean
11148:   Function GetDCOMVersion : string
11149:   Function GetMDACVersion : string
11150:   Function MarshalInterThreadInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
VarArray:OleVariant):HResult;
11151:   Function MarshalInterProcessInterfaceInStream(const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11152:   Function MarshalInterProcessInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
VarArray:OleVariant):HResult;
11153:   Function MarshalInterMachineInterfaceInStream( const iid:TIID;unk:IUnknown;var stm:IStream):HResult;
11154:   Function MarshalInterMachineInterfaceInVarArray(const iid:TIID;unk:IUnknown;var
VarArray:OleVariant):HResult;
11155:   Function CreateComponentCategory( const CatID : TGUID; const sDescription : string) : HResult
11156:   Function RegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11157:   Function UnRegisterCLSIDInCategory( const ClassID : TGUID; const CatID : TGUID) : HResult
11158:   Function ResetIStreamToStart( Stream : IStream) : Boolean
11159:   Function SizeOfIStreamContents( Stream : IStream) : Largeint
11160:   Function StreamToVariantArray( Stream : TStream) : OleVariant;
11161:   Function StreamToVariantArray1( Stream : IStream) : OleVariant;
11162:   Procedure VariantArrayToStream( VarArray : OleVariant; var Stream : TStream);
11163:   Procedure VariantArrayToStream1( VarArray : OleVariant; var Stream : IStream);
11164: end;
11165:
11166:
11167: procedure SIRegister_JclUnitConv_mX2(CL: TPSPascalCompiler);
11168: begin
11169:   Const ('CelsiusFreezingPoint', 'Extended').setExtended( 0.0);
11170:   FahrenheitFreezingPoint, 'Extended').setExtended( 32.0);
11171:   KelvinFreezingPoint, 'Extended').setExtended( 273.15);
11172:   CelsiusAbsoluteZero, 'Extended').setExtended( - 273.15);
11173:   FahrenheitAbsoluteZero, 'Extended').setExtended( - 459.67);
11174:   KelvinAbsoluteZero, 'Extended').setExtended( 0.0);
11175:   DegPerCycle, 'Extended').setExtended( 360.0);
11176:   DegPerGrad, 'Extended').setExtended( 0.9);
11177:   DegPerRad, 'Extended').setExtended( 57.295779513082320876798154814105);
11178:   GradPerCycle, 'Extended').setExtended( 400.0);
11179:   GradPerDeg, 'Extended').setExtended( 1.11111111111111111111111111111111);
11180:   GradPerRad, 'Extended').setExtended( 63.661977236758134307553505349006);
11181:   RadPerCycle, 'Extended').setExtended( 6.283185307179586476925286766559);
11182:   RadPerDeg, 'Extended').setExtended( 0.017453292519943295769236907684886);
11183:   RadPerGrad, 'Extended').setExtended( 0.015707963267948966192313216916398);
11184:   CyclePerDeg, 'Extended').setExtended( 0.0027777777777777777777777777777777);
11185:   CyclePerGrad, 'Extended').setExtended( 0.0025);
11186:   CyclePerRad, 'Extended').setExtended( 0.15915494309189533576888376337251);
11187:   ArcMinutesPerDeg, 'Extended').setExtended( 60.0);
11188:   ArcSecondsPerArcMinute, 'Extended').setExtended( 60.0);
11189:   Function HowAOneLinerCanBiteYou( const Step, Max : Longint) : Longint
11190:   Function MakePercentage( const Step, Max : Longint) : Longint

```

```

11191: Function CelsiusToKelvin( const T : double) : double
11192: Function CelsiusToFahrenheit( const T : double) : double
11193: Function KelvinToCelsius( const T : double) : double
11194: Function KelvinToFahrenheit( const T : double) : double
11195: Function FahrenheitToCelsius( const T : double) : double
11196: Function FahrenheitToKelvin( const T : double) : double
11197: Function CycleToDeg( const Cycles : double) : double
11198: Function CycleToGrad( const Cycles : double) : double
11199: Function CycleToRad( const Cycles : double) : double
11200: Function DegToCycle( const Degrees : double) : double
11201: Function DegToGrad( const Degrees : double) : double
11202: Function DegToRad( const Degrees : double) : double
11203: Function GradToCycle( const Grads : double) : double
11204: Function GradToDeg( const Grads : double) : double
11205: Function GradToRad( const Grads : double) : double
11206: Function RadToCycle( const Radians : double) : double
11207: Function RadToDeg( const Radians : double) : double
11208: Function RadToGrad( const Radians : double) : double
11209: Function DmsToDeg( const D, M : Integer; const S : double) : double
11210: Function DmsToRad( const D, M : Integer; const S : double) : double
11211: Procedure DegToDms( const Degrees : double; out D, M : Integer; out S : double)
11212: Function DegToDmsStr( const Degrees : double; const SecondPrecision : Cardinal) : string
11213: Procedure CartesianToPolar( const X, Y : double; out R, Phi : double)
11214: Procedure PolarToCartesian( const R, Phi : double; out X, Y : double)
11215: Procedure CartesianToCylinder( const X, Y, Z : double; out R, Phi, Zeta : double)
11216: Procedure CartesianToSpheric( const X, Y, Z : double; out Rho, Phi, Theta : double)
11217: Procedure CylinderToCartesian( const R, Phi, Zeta : double; out X, Y, Z : double)
11218: Procedure SphericToCartesian( const Rho, Theta, Phi : double; out X, Y, Z : double)
11219: Function CmToInch( const Cm : double) : double
11220: Function InchToCm( const Inch : double) : double
11221: Function FeetToMetre( const Feet : double) : double
11222: Function MetreToFeet( const Metre : double) : double
11223: Function YardToMetre( const Yard : double) : double
11224: Function MetreToYard( const Metre : double) : double
11225: Function NmToKm( const Nm : double) : double
11226: Function KmToNm( const Km : double) : double
11227: Function KmToSm( const Km : double) : double
11228: Function SmToKm( const Sm : double) : double
11229: Function LitreToGalUs( const Litre : double) : double
11230: Function GalUsToLitre( const GalUs : double) : double
11231: Function GalUsToGalCan( const GalUs : double) : double
11232: Function GalCanToGalUs( const GalCan : double) : double
11233: Function GalUsToGalUk( const GalUs : double) : double
11234: Function GalUkToGalUs( const GalUk : double) : double
11235: Function LitreToGalCan( const Litre : double) : double
11236: Function GalCanToLitre( const GalCan : double) : double
11237: Function LitreToGalUk( const Litre : double) : double
11238: Function GalUkToLitre( const GalUk : double) : double
11239: Function KgToLb( const Kg : double) : double
11240: Function LbToKg( const Lb : double) : double
11241: Function KgToOz( const Kg : double) : double
11242: Function OzToKg( const Oz : double) : double
11243: Function CwtUsToKg( const Cwt : double) : double
11244: Function CwtUkToKg( const Cwt : double) : double
11245: Function KaratToKg( const Karat : double) : double
11246: Function KgToCwtUs( const Kg : double) : double
11247: Function KgToCwtUk( const Kg : double) : double
11248: Function KgToKarat( const Kg : double) : double
11249: Function KgToSton( const Kg : double) : double
11250: Function KgToLton( const Kg : double) : double
11251: Function StonToKg( const STon : double) : double
11252: Function LtonToKg( const Lton : double) : double
11253: Function QrUsToKg( const Qr : double) : double
11254: Function QrUkToKg( const Qr : double) : double
11255: Function KgToQrUs( const Kg : double) : double
11256: Function KgToQrUk( const Kg : double) : double
11257: Function PascalToBar( const Pa : double) : double
11258: Function PascalToAt( const Pa : double) : double
11259: Function PascalToTorr( const Pa : double) : double
11260: Function BarToPascal( const Bar : double) : double
11261: Function AtToPascal( const At : double) : double
11262: Function TorrToPascal( const Torr : double) : double
11263: Function KnotToMs( const Knot : double) : double
11264: Function HpElectricToWatt( const HpE : double) : double
11265: Function HpMetricToWatt( const HpM : double) : double
11266: Function MsToKnot( const ms : double) : double
11267: Function WattToHpElectric( const W : double) : double
11268: Function WattToHpMetric( const W : double) : double
11269: function getBigPI: string; //PI of 1000 numbers
11270:
11271: procedure SIRegister_devcutils(CL: TPSPascalCompiler);
11272: begin
11273:   Function CDExecuteFile( const FileName, Params, DefaultDir : string; ShowCmd : Integer) : THandle
11274:   Procedure CDCopyFile( const FileName, DestName : string)
11275:   Procedure CDMoveFile( const FileName, DestName : string)
11276:   Function MakeCommaTextToColor( Text : string; Index : Integer; DefaultColor : TColor) : TColor
11277:   Procedure CDDeleteFiles( Sender : TObject; s : string)
11278:   Function CDGetTempDir : string
11279:   Function CDGetFileSize( FileName : string) : longint

```

```

11280: Function GetFileTime( FileName : string) : longint
11281: Function GetShortName( FileName : string) : string
11282: Function GetFullName( FileName : string) : string
11283: Function WinReboot : boolean
11284: Function WinDir : String
11285: Function RunFile( FileToRun : string; Params : string; Dir : string; Wait : boolean) : cardinal
11286: Function RunFile_( Cmd, WorkDir : string; Wait : boolean) : Boolean
11287: Function devExecutor : TdevExecutor
11288: end;
11289:
11290: procedure SIRegister_FileAssocs(CL: TPSPascalCompiler);
11291: begin
11292:   Procedure CheckAssociations // AssociationsCount', 'LongInt').SetInt( 7);
11293:   Procedure Associate( Index : integer)
11294:   Procedure UnAssociate( Index : integer)
11295:   Function IsAssociated( Index : integer) : boolean
11296:   Function CheckFiletype( const extension, filetype, description, verb, serverapp : string) : boolean
11297:   Procedure RegisterFiletype( const extension, filetype, description, verb, serverapp, IcoNum: string)
11298:   Procedure RegisterDDEServer( const filetype, verb, topic, servername, macro : string)
11299:   procedure RefreshIcons;
11300:   function GetShadeColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11301:   function MergColor(Colors: Array of TColor): TColor;
11302:   function NewColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11303:   procedure DimBitmap(ABitmap: TBitmap; Value: integer);
11304:   function GrayColor(ACanvas: TCanvas; clr: TColor; Value: integer): TColor;
11305:   function GetInverseColor(AColor: TColor): TColor;
11306:   procedure GrayBitmap(ABitmap: TBitmap; Value: integer);
11307:   procedure DrawBitmapShadow(B: TBitmap; ACanvas: TCanvas; X,Y: integer; ShadowColor: TColor);
11308:   procedure DrawCheckMark(ACanvas: TCanvas; X, Y: integer);
11309:   Procedure GetSystemMenuFont(Font: TFont);
11310: end;
11311:
11312: //*****unit uPSI_JvHLPParser;*****
11313: function IsStringConstant(const St: string): Boolean;
11314: function IsIntConstant(const St: string): Boolean;
11315: function IsRealConstant(const St: string): Boolean;
11316: function IsIdentifier(const ID: string): Boolean;
11317: function GetStringValue(const St: string): string;
11318: procedure ParseString(const S: string; Ss: TStrings);
11319: function IsStringConstantW(const St: WideString): Boolean;
11320: function IsIntConstantW(const St: WideString): Boolean;
11321: function IsRealConstantW(const St: WideString): Boolean;
11322: function IsIdentifierW(const ID: WideString): Boolean;
11323: function GetStringValueW(const St: WideString): WideString;
11324: procedure ParseStringW(const S: WideString; Ss: TStrings);
11325:
11326:
11327: //*****unit uPSI_JclMapi;*****
11328:
11329: Function JclSimpleSendMail( const ARecipient, AName, ASubject, ABody : string; const AAttachment :
  TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11330: Function JclSimpleSendFax( const ARecipient, AName, ASubject, ABody : string; const AAttachment :
  TFileName; ShowDialog : Boolean; AParentWND : HWND) : Boolean
11331: Function JclSimpleBringUpSendMailDialog(const ASubject, ABody: string; const
  AAttach: TFileName; AParentWND: HWND): Bool;
11332: Function MapiCheck( const Res : DWORD; IgnoreUserAbort : Boolean) : DWORD
11333: Function MapiErrorMessage( const ErrorCode : DWORD) : string
11334:
11335: procedure SIRegister_IdNTLM(CL: TPSPascalCompiler);
11336: begin
11337:   // 'Pdes_key_schedule', '^des_key_schedule // will not work
11338:   Function BuildType1Message( ADomain, AHost : String) : String
11339:   Function BuildType3Message(ADomain, AHost, AUsername: WideString; APassword, ANonce: String): String
11340:   Procedure RegisterAuthenticationMethod( MethodName : String; AuthClass : TIdAuthenticationClass)
11341:   Function FindAuthClass( AuthName : String) : TIdAuthenticationClass
11342:   GBase64CodeTable', string').SetString('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
11343:   GXXECodeTable', string').SetString('+-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
11344:   GUUECodeTable', string').SetString('!\"#$%&'()*+,-./0123456789;=<=?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
11345: end;
11346:
11347: procedure SIRegister_WDosSocketUtils(CL: TPSPascalCompiler);
11348: begin
11349:   ('IpAny', 'LongWord').SetUInt( $00000000);
11350:   IpLoopBack', 'LongWord').SetUInt( $7F000001);
11351:   IpBroadcast', 'LongWord').SetUInt( $FFFFFFF);
11352:   IpNone', 'LongWord').SetUInt( $FFFFFFF);
11353:   PortAny', 'LongWord').SetUInt( $0000);
11354:   SocketMaxConnections', 'LongInt').SetInt( 5);
11355:   TIpAddr', 'LongWord
11356:   TIpRec', 'record IpB1 : byte; IpB2 : byte; IpB3 : byte; IpB4 : Byte; end
11357:   Function HostToNetLong( HostLong : LongWord) : LongWord
11358:   Function HostToNetShort( HostShort : Word) : Word
11359:   Function NetToHostLong( NetLong : LongWord) : LongWord
11360:   Function NetToHostShort( NetShort : Word) : Word
11361:   Function StrToIp( Ip : string) : TIpAddr
11362:   Function IpToStr( Ip : TIpAddr) : string
11363: end;
11364:
11365: (*-----*)

```

```

11366: procedure SIRegister_ALSMTPClient(CL: TPSPascalCompiler);
11367: begin
11368:   TAlsmtpClientAuthType, '( AlsmtpClientAuthNone, alsmtpClientAut'
11369:   + 'hPlain, AlsmtpClientAuthLogin, AlsmtpClientAuthCramMD5, AlsmtpClientAuthCr'
11370:   + 'amShal, AlsmtpClientAuthAutoSelect )
11371:   TAlsmtpClientAuthTypeSet, 'set of TAlsmtpClientAuthType
11372: SIRegister_TAlsmtpClient(CL);
11373: end;
11374:
11375: procedure SIRegister_WDosPlcUtils(CL: TPSPascalCompiler);
11376: begin
11377:   TBitNo, 'Integer
11378:   TStByteNo, 'Integer
11379:   TStationNo, 'Integer
11380:   TInOutNo, 'Integer
11381:   TIo, '( EE, AA, NE, NA )
11382:   TBitSet, 'set of TBitNo
11383:   TAddrKind, 'set of ( akBit0, akBit1, akBit2, akOut, akNot, akBus )
11384:   TBitAddrRec, 'record Kind : TAddrKind; InOutNo : TInOutNo; ByteNo : Byte; end
11385:   TBitAddr, 'LongInt
11386:   TByteAddrRec, 'record Kind : TAddrKind; ByteNo : Byte; end
11387:   TByteAddr, 'SmallInt
11388:   TInOutState, '( iosInit, iosHalt, iosRun, iosError )
11389:   Function BitAddr(aIo: TIo; aInOutNo : TInOutNo; aByteNo : Byte; aBitNo : TBitNo) : TBitAddr
11390:   Function BusBitAddr(aIo:TIo;aInOutNo:TInOutNo;aStat:TStationNo;aStByteNo:TStByteNo;aBitNo:TBitNo) :
   TBitAddr;
11391:   Procedure BitAddrToValues(aBitAdr:TBitAdr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte;var
   aBitNo:TBitNo);
11392:   Function BitAddrToStr( Value : TBitAddr) : string
11393:   Function StrToBitAddr( const Value : string) : TBitAddr
11394:   Function ByteAddr( aIo : TIo; aInOutNo : TInOutNo; aByteNo : Byte) : TByteAddr
11395:   Function BusByteAddr(aIo:TIo;aInOutNo:TInOutNo;aStation:TStationNo;aStByteNo: TStByteNo):TByteAddr
11396:   Procedure ByteAddrToValues(aByteAddr:TByteAddr;var aIo:TIo;var aInOutNo:TInOutNo;var aByteNo:Byte)
11397:   Function ByteAddrToStr( Value : TByteAddr) : string
11398:   Function StrToByteAddr( const Value : string) : TByteAddr
11399:   Procedure IncByteAddr( var ByteAddr : TByteAddr; Increment : Integer)
11400:   Procedure DecByteAddr( var ByteAddr : TByteAddr; Decrement : Integer)
11401:   Function InOutStateToStr( State : TInOutState) : string
11402:   Function MasterErrorToStr( ErrorCode : TErrorCode) : string
11403:   Function SlaveErrorToStr( ErrorCode : TErrorCode) : string
11404: end;
11405:
11406: procedure SIRegister_WDosTimers(CL: TPSPascalCompiler);
11407: begin
11408:   TIntFreq, '( ifNone, if32768, if16384, if8192, if4096, if2048, '
11409:   + 'if1024, if512, if256, if128, if64, if32, if16, if8, if4, if2 )
11410:   DpmiPmVector, 'Int64
11411:   'DInterval','LongInt').SetInt( 1000);
11412:   //'DEnabled','Boolean')BoolToStr( True);
11413:   'DIntFreq','string').SetString( if64
11414:   //'DMessages','Boolean').SetString( if64);
11415:   SIRegister_TwdxCustomTimer(CL);
11416:   SIRegister_TwdxTimer(CL);
11417:   SIRegister_TwdxRtcTimer(CL);
11418:   SIRegister_TCustomIntTimer(CL);
11419:   SIRegister_TIntTimer(CL);
11420:   SIRegister_TRtcIntTimer(CL);
11421:   Function RealNow : TDateTime
11422:   Function MsToDateTime( MilliSecond : LongInt) : TDateTime
11423:   Function DateTimeToMs( Time : TDateTime) : LongInt
11424: end;
11425:
11426: procedure SIRegister_IdSysLogMessage(CL: TPSPascalCompiler);
11427: begin
11428:   TIdSyslogPRI, 'Integer
11429:   TIdSyslogFacility, '( sfKernel, sfUserLevel, sfMailSystem, sfSy'
11430:   + 'stemDaemon, sfSecurityOne, sfSysLogInternal, sfLPR, sfNNTP, sfUUCP, sfCloc'
11431:   + 'kDaemonOne, sfSecurityTwo, sfTPDaemon, sfNTP, sfLogAudit, sfLogAlert, sfC'
11432:   + 'lockDaemonTwo, sfLocalUseZero, sfLocalUseOne, sfLocalUseTwo, sfLocalUseThr'
11433:   + 'ee, sfLocalUseFour, sfLocalUseFive, sfLocalUseSix, sfLocalUseSeven )
11434:   TIdSyslogSeverity, '( slEmergency, slAlert, slCritical, slError'
11435:   + ', slWarning, slNotice, slInformational, slDebug )
11436:   SIRegister_TIdSysLogMsgPart(CL);
11437:   SIRegister_TIdSysLogMessage(CL);
11438:   Function FacilityToString( AFac : TIdSyslogFacility) : string
11439:   Function SeverityToString( ASec : TIdSyslogSeverity) : string
11440:   Function NoToSeverity( ASev : Word) : TIdSyslogSeverity
11441:   Function logSeverityToNo( ASev : TIdSyslogSeverity) : Word
11442:   Function NoToFacility( AFac : Word) : TIdSyslogFacility
11443:   Function logFacilityToNo( AFac : TIdSyslogFacility) : Word
11444: end;
11445:
11446: procedure SIRegister_TextUtils(CL: TPSPascalCompiler);
11447: begin
11448:   'UWhitespace','String').SetString( '(:\s*)
11449:   Function StripSpaces( const AText : string) : string
11450:   Function CharCount( const AText : string; Ch : Char) : Integer
11451:   Function BalancedText( const AText : string; const Ch1, Ch2 : Char; const Count : Integer) : string
11452:   Function BalancedTextReg( const AText:string; const Ch1, Ch2 : Char; const Count : Integer) : string

```



```

11453: end;
11454:
11455:
11456: procedure SIRegister_ExtPascalUtils(CL: TPSPascalCompiler);
11457: begin
11458:   ExtPascalVersion('String').SetString( '0.9.8
11459:   AddTypeS('TBrowser', '( brUnknown, brIE, brFirefox, brChrome, brSafari, br'
11460:   +'Opera, brKonqueror, brMobileSafari )
11461:   AddTypeS('TCSSUnit', '( cssPX, cssPerc, cssEM, cssEX, cssIN, cssCM, cssMM, cssPT, cssPC, cssnone )
11462:   AddTypeS('TextProcedure', 'Procedure
11463:   Function DetermineBrowser( const UserAgentStr : string ) : TBrowser
11464:   Function ExtExtract(const Delims:array of string;var S:string;var Matches:TStringList;Remove:bool):bool;
11465:   Function Explode( Delim : char; const S : string; Separator : char ) : TStringList
11466:   Function FirstDelimiter( const Delimiters, S : string; Offset : integer ) : integer
11467:   Function RPosEx( const Substr, Str : string; Offset : integer ) : integer
11468:   Function CountStr( const Substr, Str : string; UntilStr : string ) : integer
11469:   Function StrToJS( const S : string; UseBR : boolean ) : string
11470:   Function CaseOf( const S : string; const Cases : array of string ) : integer
11471:   Function RCaseOf( const S : string; const Cases : array of string ) : integer
11472:   Function EnumToJSString( TypeInfo : PTypeInfo; Value : integer ) : string
11473:   Function SetPaddings(Top:integer;Right:int;Bottom:intr;Left:integer;CSSUnit:TCSSUnit;Header:bool):string;
11474:   Function SetMargins(Top:integer;Right:int;Bottom:int;Left:integer;CSSUnit:TCSSUnit;Header:bool): string;
11475:   Function ExtBefore( const BeforeS, AfterS, S : string ) : boolean
11476:   Function IsUpperCase( S : string ) : boolean
11477:   Function BeautifyJS(const AScript:string;const StartingLevel:integer;SplitHTMLNewLine: boolean):string;
11478:   Function BeautifyCSS( const AStyle : string ) : string
11479:   Function LengthRegExp( Rex : string; CountAll : Boolean ) : integer
11480:   Function JSDateToDateTime( JSDate : string ) : TDateTime
11481: end;
11482:
11483: procedure SIRegister_JclShell(CL: TPSPascalCompiler);
11484: begin
11485:   TSHDeleteOption', '( doSilent, doAllowUndo, doFilesOnly )
11486:   TSHDeleteOptions', 'set of TSHDeleteOption
11487:   TSHRenameOption', '( roSilent, roRenameOnCollision )
11488:   TSHRenameOptions', 'set of TSHRenameOption
11489:   Function SHDeleteFiles( Parent : HWND; const Files : string; Options : TSHDeleteOptions ) : Boolean
11490:   Function SHDeleteFolder( Parent : HWND; const Folder : string; Options : TSHDeleteOptions ) : Boolean
11491:   Function SHRenameFile( const Src, Dest : string; Options : TSHRenameOptions ) : Boolean
11492:   TEnumFolderFlag', '( efFolders, efNonFolders, efIncludeHidden )
11493:   TEnumFolderFlags', 'set of TEnumFolderFlag
11494:   TEnumFolderRec', 'record DisplayName : string; Attributes : DWOR'
11495:   +'D; IconLarge : HICON; IconSmall : HICON; Item : PItemIdList; EnumIdList : '
11496:   +'IEnumIdList; Folder : IShellFolder; end
11497:   Function SHEnumFolderFirst(const Folder:string;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Boolean;
11498:   Function SHEnumSpecialFolderFirst(SpecialFolder:DWORD;Flags:TEnumFolderFlags;var F:TEnumFolderRec):Bool;
11499:   Procedure SHEnumFolderClose( var F : TEnumFolderRec)
11500:   Function SHEnumFolderNext( var F : TEnumFolderRec ) : Boolean
11501:   Function GetSpecialFolderLocation( const Folder : Integer ) : string
11502:   Function DisplayPropDialog( const Handle : HWND; const FileName : string ) : Boolean;
11503:   Function DisplayPropDialog1( const Handle : HWND; const Item : PItemIdList ) : Boolean;
11504:   Function DisplayContextMenu( const Handle : HWND; const FileName : string; Pos : TPoint ) : Boolean
11505:   Function OpenFolder( const Path : string; Parent : HWND ) : Boolean
11506:   Function OpenSpecialFolder( FolderID : Integer; Parent : HWND ) : Boolean
11507:   Function SHReallocMem( var P : Pointer; Count : Integer ) : Boolean
11508:   Function SHAllocMem( out P : Pointer; Count : Integer ) : Boolean
11509:   Function SHGetMem( var P : Pointer; Count : Integer ) : Boolean
11510:   Function SHFreeMem( var P : Pointer ) : Boolean
11511:   Function DriveToPidlBind( const DriveName : string; out Folder : IShellFolder ) : PItemIdList
11512:   Function PathToPidl( const Path : string; Folder : IShellFolder ) : PItemIdList
11513:   Function PathToPidlBind( const FileName : string; out Folder : IShellFolder ) : PItemIdList
11514:   Function PidlBindToParent(const IdList:PItemIdList;out Folder:IShellFolder;out Last:PItemIdList):Bool;
11515:   Function PidlCompare( const Pidl1, Pidl2 : PItemIdList ) : Boolean
11516:   Function PidlCopy( const Source : PItemIdList; out Dest : PItemIdList ) : Boolean
11517:   Function PidlFree( var IdList : PItemIdList ) : Boolean
11518:   Function PidlGetDepth( const Pidl : PItemIdList ) : Integer
11519:   Function PidlGetLength( const Pidl : PItemIdList ) : Integer
11520:   Function PidlGetNext( const Pidl : PItemIdList ) : PItemIdList
11521:   Function PidlToPath( IdList : PItemIdList ) : string
11522:   Function StrRetFreeMem( StrRet : TStrRet ) : Boolean
11523:   Function StrRetToString( IdList : PItemIdList; StrRet : TStrRet; Free : Boolean ) : string
11524:   PShellLink', '^TShellLink // will not work
11525:   TShellLink', 'record Arguments : string; ShowCmd : Integer; Work'
11526:   +'ingDirectory : string; IdList : PItemIdList; Target : string; Description '
11527:   +' : string; IconLocation : string; IconIndex : Integer; HotKey : Word; end
11528:   Procedure ShellLinkFree( var Link : TShellLink)
11529:   Function ShellLinkResolve( const FileName : string; var Link : TShellLink ) : HRESULT
11530:   Function ShellLinkCreate( const Link : TShellLink; const FileName : string ) : HRESULT
11531:   Function ShellLinkCreateSystem(const Link:TShellLink;const Folder:Integer; const FileName:string):HRESULT;
11532:   Function ShellLinkGetIcon( const Link : TShellLink; const Icon : TIcon ) : Boolean
11533:   Function SHDllGetVersion( const FileName : string; var Version : TDllVersionInfo ) : Boolean
11534:   Function GetSystemIcon( IconIndex : Integer; Flags : Cardinal ) : HICON
11535:   Function OverlayIcon( var Icon : HICON; Overlay : HICON; Large : Boolean ) : Boolean
11536:   Function OverlayIconShortCut( var Large, Small : HICON ) : Boolean
11537:   Function OverlayIconShared( var Large, Small : HICON ) : Boolean
11538:   Function SHGetItemInfoTip( const Folder : IShellFolder; Item : PItemIdList ) : string
11539:   Function ShellExecEx(const FileName:string;const Parameters:string;const Verb:string; CmdShow:Int):Bool;
11540:   Function ShellExec(Wnd: Integer;const Operation,FileName,Parameters,Directy:string;ShowCommand:Int):Bool;
11541:   Function ShellExecAndWait(const FileName:string;const Paramets:string;const Verb:string;CmdShow:Int):Bool;

```

```

11542: Function ShellOpenAs( const FileName : string) : Boolean
11543: Function ShellRasDial( const EntryName : string) : Boolean
11544: Function ShellRunControlPanel( const NameOrFileName:string; AppletNumber:Integer):Boolean
11545: Function GetFileNameIcon( const FileName : string; Flags : Cardinal) : HICON
11546: TJclFileExeType, '( etError, etMsDos, etWin16, etWin32Gui, etWin32Con )
11547: Function GetFileExeType( const FileName : TFileName) : TJclFileExeType
11548: Function ShellFindExecutable( const FileName, DefaultDir : string) : string
11549: Procedure keybd_event( bVk : Byte; bScan : Byte; dwFlags, dwExtraInfo : DWORD)
11550: Function OemKeyScan( wOemChar : Word) : DWORD
11551: Procedure mouse_event( dwFlags, dx, dy, dwData, dwExtraInfo : DWORD)
11552: end;
11553:
11554: procedure SIRegister_cXMLFunctions(CL: TPSPascalCompiler);
11555: begin
11556:   xmlVersion','string').SetString( '1.0 FindClass('TOBJECT'),'Exml
11557:   //Function xmlValidChar( const Ch : AnsiChar) : Boolean;
11558:   Function xmlValidChar1( const Ch : UCS4Char) : Boolean;
11559:   Function xmlValidChar2( const Ch : WideChar) : Boolean;
11560:   Function xmlIsSpaceChar( const Ch : WideChar) : Boolean
11561:   Function xmlIsLetter( const Ch : WideChar) : Boolean
11562:   Function xmlIsDigit( const Ch : WideChar) : Boolean
11563:   Function xmlIsNameStartChar( const Ch : WideChar) : Boolean
11564:   Function xmlIsNameChar( const Ch : WideChar) : Boolean
11565:   Function xmlIsPubidChar( const Ch : WideChar) : Boolean
11566:   Function xmlValidName( const Text : UnicodeString) : Boolean
11567:   //xmlSpace','Char').SetString( #$20 or #$9 or #$D or #$A);
11568:   //Function xmlSkipSpace( var P : PWideChar) : Boolean
11569:   //Function xmlSkipEq( var P : PWideChar) : Boolean
11570:   //Function xmlExtractQuotedText( var P : PWideChar; var S : UnicodeString) : Boolean
11571:   //Function xmlGetEntityEncoding( const Buf : Pointer; const BufSize : Integer; out HeaderSize : Integer)
: TUnicodeCodecClass
11572:   Function xmlResolveEntityReference( const RefName : UnicodeString) : WideChar
11573:   Function xmlTag( const Tag : UnicodeString) : UnicodeString
11574:   Function xmlEndTag( const Tag : UnicodeString) : UnicodeString
11575:   Function xmlAttrTag( const Tag : UnicodeString; const Attr : UnicodeString) : UnicodeString
11576:   Function xmlEmptyTag( const Tag, Attr : UnicodeString) : UnicodeString
11577:   Procedure xmlSafeTextInPlace( var Txt : UnicodeString)
11578:   Function xmlSafeText( const Txt : UnicodeString) : UnicodeString
11579:   Function xmlSpaceIndent( const IndentLength : Integer; const IndentLevel : Integer):UnicodeString
11580:   Function xmlTabIndent( const IndentLevel : Integer) : UnicodeString
11581:   Function xmlComment( const Comment : UnicodeString) : UnicodeString
11582:   Procedure SelfTestcXMLFunctions
11583: end;
11584:
11585: (*-----*)
11586: procedure SIRegister_DepWalkUtils(CL: TPSPascalCompiler);
11587: begin
11588:   Function AwaitCursor : IUnknown
11589:   Function ChangeCursor( NewCursor : TCursor) : IUnknown
11590:   Procedure SuspendRedraw( AControl : TWinControl; Suspend : boolean)
11591:   Function YesNo( const ACaption, AMsg : string) : boolean
11592:   Procedure strTokenize( const S : string; Delims : TSysCharSet; Results : TStrings)
11593:   Function GetBorlandLibPath( Version : integer; ForDelphi : boolean) : string
11594:   Function GetExpandedLibRoot( Version : integer; ForDelphi : boolean) : string
11595:   Procedure GetPathList( Version : integer; ForDelphi : boolean; Strings : TStrings)
11596:   Procedure GetSystemPaths( Strings : TStrings)
11597:   Procedure MakeEditNumeric( EditHandle : integer)
11598: end;
11599:
11600: procedure SIRegister_yuvconverts(CL: TPSPascalCompiler);
11601: begin
11602:   AddTypes('TVideoCodec', '(vcUnknown,vcRGB,vcYUY2,vcUYVY,vcBTYUV,vcYV,U9,vcYUV12,vcY8,vcY211)
11603:   'BI_YUY2','LongWord').SetUInt( $32595559);
11604:   'BI_UYVY','LongWord').SetUInt( $59565955);
11605:   'BI_BTUV','LongWord').SetUInt( $50313459);
11606:   'BI_YVU9','LongWord').SetUInt( $39555659);
11607:   'BI_YUV12','LongWord').SetUInt( $30323449);
11608:   'BI_Y8','LongWord').SetUInt( $20203859);
11609:   'BI_Y211','LongWord').SetUInt( $31313259);
11610:   Function BICompressionToVideoCodec( Value : DWord) : TVideoCodec
11611:   Function ConvertCodecToRGB(CoDec:TVideoCodec;Src,Dst:Pointer;AWidth,AHeight:Integer):Boolean;
11612: end;
11613:
11614: (*-----*)
11615: procedure SIRegister_AviCap(CL: TPSPascalCompiler);
11616: begin
11617:   'WM_USER','LongWord').SetUInt( $0400);
11618:   'WM_CAP_START','LongWord').SetUInt($0400);
11619:   'WM_CAP_END','longword').SetUInt($0400+85);
11620:   //WM_CAP_START+ 85
11621:   // WM_CAP_SET_CALLBACK_CAPCONTROL = (WM_CAP_START+ 85);
11622:   Function capSetCallbackOnError( hwnd : THandle; fpProc : LongInt) : LongInt
11623:   Function capSetCallbackOnStatus( hwnd : THandle; fpProc : LongInt) : LongInt
11624:   Function capSetCallbackOnYield( hwnd : THandle; fpProc : LongInt) : LongInt
11625:   Function capSetCallbackOnFrame( hwnd : THandle; fpProc : LongInt) : LongInt
11626:   Function capSetCallbackOnVideoStream( hwnd : THandle; fpProc : LongInt) : LongInt
11627:   Function capSetCallbackOnWaveStream( hwnd : THandle; fpProc : LongInt) : LongInt
11628:   Function capSetCallbackOnCapControl( hwnd : THandle; fpProc : LongInt) : LongInt
11629:   Function capSetUserData( hwnd : THandle; lUser : LongInt) : LongInt

```

```

11630: Function capGetUserData( hwnd : THandle) : LongInt
11631: Function capDriverConnect( hwnd : THandle; I : Word) : LongInt
11632: Function capDriverDisconnect( hwnd : THandle) : LongInt
11633: Function capDriverGetName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11634: Function capDriverGetVersion( hwnd : THandle; szVer : LongInt; wSize : Word) : LongInt
11635: Function capDriverGetCaps( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11636: Function capFileSetCaptureFile( hwnd : THandle; szName : LongInt) : LongInt
11637: Function capFileGetCaptureFile( hwnd : THandle; szName : LongInt; wSize : Word):LongInt
11638: Function capFileAlloc( hwnd : THandle; dwSize : LongInt) : LongInt
11639: Function capFileSaveAs( hwnd : THandle; szName : LongInt) : LongInt
11640: Function capFileSetInfoChunk( hwnd : THandle; lpInfoChunk : LongInt) : LongInt
11641: Function capFileSaveDIB( hwnd : THandle; szName : LongInt) : LongInt
11642: Function capEditCopy( hwnd : THandle) : LongInt
11643: Function capSetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11644: Function capGetAudioFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11645: Function capGetVideoFormatSize( hwnd : THandle) : LongInt
11646: Function capDlgVideoFormat( hwnd : THandle) : LongInt
11647: Function capDlgVideoSource( hwnd : THandle) : LongInt
11648: Function capDlgVideoDisplay( hwnd : THandle) : LongInt
11649: Function capDlgVideoCompression( hwnd : THandle) : LongInt
11650: Function capGetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11651: Function capGetVideoFormatSize( hwnd : THandle) : LongInt
11652: Function capSetVideoFormat( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11653: Function capPreview( hwnd : THandle; f : Word) : LongInt
11654: Function capPreviewRate( hwnd : THandle; wMS : Word) : LongInt
11655: Function capOverlay( hwnd : THandle; f : Word) : LongInt
11656: Function capPreviewScale( hwnd : THandle; f : Word) : LongInt
11657: Function capGetStatus( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11658: Function capSetScrollPos( hwnd : THandle; lpP : LongInt) : LongInt
11659: Function capGrabFrame( hwnd : THandle) : LongInt
11660: Function capGrabFrameNoStop( hwnd : THandle) : LongInt
11661: Function capCaptureSequence( hwnd : THandle) : LongInt
11662: Function capCaptureSequenceNoFile( hwnd : THandle) : LongInt
11663: Function capCaptureStop( hwnd : THandle) : LongInt
11664: Function capCaptureAbort( hwnd : THandle) : LongInt
11665: Function capCaptureSingleFrameOpen( hwnd : THandle) : LongInt
11666: Function capCaptureSingleFrameClose( hwnd : THandle) : LongInt
11667: Function capCaptureSingleFrame( hwnd : THandle) : LongInt
11668: Function capCaptureGetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11669: Function capCaptureSetSetup( hwnd : THandle; s : LongInt; wSize : Word) : LongInt
11670: Function capSetMCIDeviceName( hwnd : THandle; szName : LongInt) : LongInt
11671: Function capGetMCIDeviceName( hwnd : THandle; szName : LongInt; wSize : Word) : LongInt
11672: Function capPaletteOpen( hwnd : THandle; szName : LongInt) : LongInt
11673: Function capPaletteSave( hwnd : THandle; szName : LongInt) : LongInt
11674: Function capPalettePaste( hwnd : THandle) : LongInt
11675: Function capPaletteAuto( hwnd : THandle; iFrames : Word; iColors : LongInt) : LongInt
11676: Function capPaletteManual( hwnd : THandle; fGrab : Word; iColors : LongInt) : LongInt
11677: //PCapDriverCaps', '^TCapDriverCaps // will not work
11678: TCapDriverCaps', 'record wDeviceIndex : WORD; fHasOverlay : BOOL'
11679: + 'fHasDlgVideoSource : BOOL; fHasDlgVideoFormat : BOOL; fHasDlgVideoDispla'
11680: + 'y : BOOL; fCaptureInitialized : BOOL; fDriverSuppliesPalettes : BOOL; hVid'
11681: + 'eoIn : THandle; hVideoOut : THandle; hVideoExtIn:THandle; hVideoExtOut:THandle; end'
11682: //PCapStatus', '^TCapStatus // will not work
11683: TCapStatus', 'record uiImageWidth : UINT; uiImageHeight : UINT; '
11684: + 'fLiveWindow : BOOL; fOverlayWindow : BOOL; fScale : BOOL; ptScroll : TPOINT'
11685: + 'T; fUsingDefaultPalette : BOOL; fAudioHardware : BOOL; fCapFileExists : BO'
11686: + 'OL; dwCurrentVideoFrame : DWORD; dwCurrentVideoFramesDropped : DWORD; dwCu'
11687: + 'rrentWaveSamples : DWORD; dwCurrentTimeElapsedMS : DWORD; hPalCurrent : HP'
11688: + 'ALETTE; fCapturingNow : BOOL; dwReturn : DWORD; wNumVideoAllocated : WORD;'
11689: + 'wNumAudioAllocated : WORD; end'
11690: //PCaptureParms', '^TCaptureParms // will not work
11691: TCaptureParms', 'record dwRequestMicroSecPerFrame : DWORD; fMake'
11692: + 'UserHitOKToCapture : BOOL; wPercentDropForError : WORD; fYield : BOOL; dwI'
11693: + 'ndexSize : DWORD; wChunkGranularity : WORD; fUsingDOSMemory : BOOL; wNumVi'
11694: + 'deoRequested : WORD; fCaptureAudio : BOOL; wNumAudioRequested : WORD; vKey'
11695: + 'Abort : WORD; fAbortLeftMouse : BOOL; fAbortRightMouse : BOOL; fLimitEnabl'
11696: + 'ed : BOOL; wTimeLimit : WORD; fMCIControl : BOOL; fStepMCIDevice : BOOL; d'
11697: + 'wMCIStartTime : DWORD; dwMCIStopTime : DWORD; fStepCaptureAt2x : BOOL; wSt'
11698: + 'epCaptureAverageFrames : WORD; dwAudioBufferSize : DWORD; fDisableWriteCac'
11699: + 'he : BOOL; AVStreamMaster : WORD; end'
11700: // PCapInfoChunk', '^TCapInfoChunk // will not work
11701: //TCapInfoChunk', 'record fccInfoID : FOURCC; lpData : LongInt; cbData : LongInt; end
11702: 'CONTROLCALLBACK_PREROLL', 'LongInt').SetInt( 1);
11703: 'CONTROLCALLBACK_CAPTURING', 'LongInt').SetInt( 2);
11704: Function capCreateCaptureWindow( lpszWindowName: PChar; dwStyle : DWORD; x, y : Integer; nWidth, nHeight
: Integer; hwndParent : THandle; nID : Integer) : THandle
11705: Function
capGetDriverDescription(wDriverIndex:DWORD;lpszName:PChar;cbName:Integer;lpszVer:PChar;cbVer:Int):Bool;
11706: 'IDS_CAP_BEGIN', 'LongInt').SetInt( 300);
11707: 'IDS_CAP_END', 'LongInt').SetInt( 301);
11708: 'IDS_CAP_INFO', 'LongInt').SetInt( 401);
11709: 'IDS_CAP_OUTOFMEM', 'LongInt').SetInt( 402);
11710: 'IDS_CAP_FILEEXISTS', 'LongInt').SetInt( 403);
11711: 'IDS_CAP_ERRORPALOPEN', 'LongInt').SetInt( 404);
11712: 'IDS_CAP_ERRORPALSAVE', 'LongInt').SetInt( 405);
11713: 'IDS_CAP_ERRORDIBSAVE', 'LongInt').SetInt( 406);
11714: 'IDS_CAP_DEFAVIEXT', 'LongInt').SetInt( 407);
11715: 'IDS_CAP_DEFPALEXT', 'LongInt').SetInt( 408);
11716: 'IDS_CAP_CANTOPEN', 'LongInt').SetInt( 409);

```

```

11717: 'IDS_CAP_SEQ_MSGSTART', 'LongInt').SetInt( 410);
11718: 'IDS_CAP_SEQ_MSGSTOP', 'LongInt').SetInt( 411);
11719: 'IDS_CAP_VIDEDITERR', 'LongInt').SetInt( 412);
11720: 'IDS_CAP_READONLYFILE', 'LongInt').SetInt( 413);
11721: 'IDS_CAP_WRITEERROR', 'LongInt').SetInt( 414);
11722: 'IDS_CAP_NODISKSPACE', 'LongInt').SetInt( 415);
11723: 'IDS_CAP_SETFILESIZE', 'LongInt').SetInt( 416);
11724: 'IDS_CAP_SAVEASPERCENT', 'LongInt').SetInt( 417);
11725: 'IDS_CAP_DRIVER_ERROR', 'LongInt').SetInt( 418);
11726: 'IDS_CAP_WAVE_OPEN_ERROR', 'LongInt').SetInt( 419);
11727: 'IDS_CAP_WAVE_ALLOC_ERROR', 'LongInt').SetInt( 420);
11728: 'IDS_CAP_WAVE_PREPARE_ERROR', 'LongInt').SetInt( 421);
11729: 'IDS_CAP_WAVE_ADD_ERROR', 'LongInt').SetInt( 422);
11730: 'IDS_CAP_WAVE_SIZE_ERROR', 'LongInt').SetInt( 423);
11731: 'IDS_CAP_VIDEO_OPEN_ERROR', 'LongInt').SetInt( 424);
11732: 'IDS_CAP_VIDEO_ALLOC_ERROR', 'LongInt').SetInt( 425);
11733: 'IDS_CAP_VIDEO_PREPARE_ERROR', 'LongInt').SetInt( 426);
11734: 'IDS_CAP_VIDEO_ADD_ERROR', 'LongInt').SetInt( 427);
11735: 'IDS_CAP_VIDEO_SIZE_ERROR', 'LongInt').SetInt( 428);
11736: 'IDS_CAP_FILE_OPEN_ERROR', 'LongInt').SetInt( 429);
11737: 'IDS_CAP_FILE_WRITE_ERROR', 'LongInt').SetInt( 430);
11738: 'IDS_CAP_RECORDING_ERROR', 'LongInt').SetInt( 431);
11739: 'IDS_CAP_RECORDING_ERROR2', 'LongInt').SetInt( 432);
11740: 'IDS_CAP_AVI_INIT_ERROR', 'LongInt').SetInt( 433);
11741: 'IDS_CAP_NO_FRAME_CAP_ERROR', 'LongInt').SetInt( 434);
11742: 'IDS_CAP_NO_PALETTE_WARN', 'LongInt').SetInt( 435);
11743: 'IDS_CAP_MCI_CONTROL_ERROR', 'LongInt').SetInt( 436);
11744: 'IDS_CAP_MCI_CANT_STEP_ERROR', 'LongInt').SetInt( 437);
11745: 'IDS_CAP_NO_AUDIO_CAP_ERROR', 'LongInt').SetInt( 438);
11746: 'IDS_CAP_AVI_DRAWDIB_ERROR', 'LongInt').SetInt( 439);
11747: 'IDS_CAP_COMPRESSOR_ERROR', 'LongInt').SetInt( 440);
11748: 'IDS_CAP_AUDIO_DROP_ERROR', 'LongInt').SetInt( 441);
11749: 'IDS_CAP_STAT_LIVE_MODE', 'LongInt').SetInt( 500);
11750: 'IDS_CAP_STAT_OVERLAY_MODE', 'LongInt').SetInt( 501);
11751: 'IDS_CAP_STAT_CAP_INIT', 'LongInt').SetInt( 502);
11752: 'IDS_CAP_STAT_CAP_FINI', 'LongInt').SetInt( 503);
11753: 'IDS_CAP_STAT_PALETTE_BUILD', 'LongInt').SetInt( 504);
11754: 'IDS_CAP_STAT_OPTPAL_BUILD', 'LongInt').SetInt( 505);
11755: 'IDS_CAP_STAT_I_FRAMES', 'LongInt').SetInt( 506);
11756: 'IDS_CAP_STAT_L_FRAMES', 'LongInt').SetInt( 507);
11757: 'IDS_CAP_STAT_CAP_L_FRAMES', 'LongInt').SetInt( 508);
11758: 'IDS_CAP_STAT_CAP_AUDIO', 'LongInt').SetInt( 509);
11759: 'IDS_CAP_STAT_VIDEOCURRENT', 'LongInt').SetInt( 510);
11760: 'IDS_CAP_STAT_VIDEOAUDIO', 'LongInt').SetInt( 511);
11761: 'IDS_CAP_STAT_VIDEOONLY', 'LongInt').SetInt( 512);
11762: 'IDS_CAP_STAT_FRAMESDROPPED', 'LongInt').SetInt( 513);
11763: 'AVICAP32', 'String').SetString( 'AVICAP32.dll
11764: end;
11765:
11766: procedure SIRegister_ALFcnMisc(CL: TPSPascalCompiler);
11767: begin
11768:   Function ALBoolToInt( Value : Boolean ) : Integer
11769:   Function ALMediumPos( LTotal, LBorder, LObject : integer ) : Integer
11770:   Function ALIsValidEmail( const Value : AnsiString ) : boolean
11771:   Function ALLocalDateTimeToGMTDateTime( const aLocalDateTime : TDateTime ) : TDateTime
11772:   Function ALInc( var x : integer; Count : integer ) : Integer
11773:   function ALCopyStr(const aSourceString: AnsiString; aStart, aLength: Integer): AnsiString
11774:   function ALGetStringFromFile(filename: AnsiString; const ShareMode: Word = fmShareDenyWrite): AnsiString;
11775:   procedure ALSaveStringToFile(Str: AnsiString; filename: AnsiString);
11776:   Function ALIsInteger(const S: AnsiString): Boolean;
11777:   function ALIsDecimal(const S: AnsiString): boolean;
11778:   Function ALStringToWideString(const S: AnsiString; const aCodePage: Word): WideString;
11779:   function ALWideStringToString(const WS: WideString; const aCodePage: Word): AnsiString;
11780:   function ALQuotedStr(const S: AnsiString; const Quote: AnsiChar = ''''): AnsiString;
11781:   function ALDequotedStr(const S: AnsiString; AQuote: AnsiChar): AnsiString;
11782:   function ALUTF8removeBOM(const S: AnsiString): AnsiString;
11783:   Function ALRandomStr1(const aLength: Longint; const aCharset: Array of Char): AnsiString;
11784:   Function ALRandomStr(const aLength: Longint): AnsiString;
11785:   Function ALRandomStrU1(const aLength: Longint; const aCharset: Array of Char): String;
11786:   Function ALRandomStrU(const aLength: Longint): String;
11787: end;
11788:
11789: procedure SIRegister_ALJSONDoc(CL: TPSPascalCompiler);
11790: begin
11791:   Procedure ALJSONToTStrings(const AJsonStr: AnsiString; aLst: TALStrings; const aNullStr: AnsiString; const
aTrueStr: AnsiString; const aFalseStr : AnsiString)
11792: end;
11793:
11794: procedure SIRegister_ALWindows(CL: TPSPascalCompiler);
11795: begin
11796:   _ALMEMORYSTATUSEX', 'record dwLength : DWORD; dwMemoryLoad : DWO'
11797:   + 'RD; ullTotalPhys : Int64; ullAvailPhys : Int64; ullTotalPageFile : Int64; '
11798:   + 'ullAvailPageFile : Int64; ullTotalVirtual : Int64; ullAvailVirtual : Int64'
11799:   + 'ullAvailExtendedVirtual : Int64; end
11800:   TALMemoryStatusEx', '_ALMEMORYSTATUSEX
11801:   Function ALGlobalMemoryStatusEx( var lpBuffer : TALMEMORYSTATUSEX ) : BOOL
11802:   Function ALInterlockedExchange64( var Target : LONGLONG; Value : LONGLONG ) : LONGLONG
11803:   'INVALID_SET_FILE_POINTER', 'LongInt').SetInt( DWORD ( - 1 ));
11804:   'QUOTA_LIMITS_HARDWS_MIN_DISABLE', 'LongWord').SetUInt( $2);

```



```

11805: 'QUOTA_LIMITS_HARDWS_MIN_ENABLE','LongWord').SetUInt( $1);
11806: 'QUOTA_LIMITS_HARDWS_MAX_DISABLE','LongWord').SetUInt( $8);
11807: 'QUOTA_LIMITS_HARDWS_MAX_ENABLE','LongWord').SetUInt( $4);
11808: end;
11809:
11810: procedure SIRegister_IPCThrd(CL: TPSPascalCompiler);
11811: begin
11812:   SIRegister_THandledObject(CL);
11813:   SIRegister_TEvent(CL);
11814:   SIRegister_TMutex(CL);
11815:   SIRegister_TSharedMem(CL);
11816:   'TRACE_BUF_SIZE','LongInt').SetInt( 200 * 1024);
11817:   'TRACE_BUFFER','String').SetString( 'TRACE_BUFFER
11818:   'TRACE_MUTEX','String').SetString( 'TRACE_MUTEX
11819:   //PTraceEntry', '^TTraceEntry // will not work
11820:   SIRegister_TIPCTracer(CL);
11821:   'MAX_CLIENTS','LongInt').SetInt( 6);
11822:   'IPCTIMEOUT','LongInt').SetInt( 2000);
11823:   'IPCBUFFER_NAME','String').SetString( 'BUFFER_NAME
11824:   'BUFFER_MUTEX_NAME','String').SetString( 'BUFFER_MUTEX
11825:   'MONITOR_EVENT_NAME','String').SetString( 'MONITOR_EVENT
11826:   'CLIENT_EVENT_NAME','String').SetString( 'CLIENT_EVENT
11827:   'CONNECT_EVENT_NAME','String').SetString( 'CONNECT_EVENT
11828:   'CLIENT_DIR_NAME','String').SetString( 'CLIENT_DIRECTORY
11829:   'CLIENT_DIR_MUTEX','String').SetString( 'DIRECTORY_MUTEX
11830:   FindClass('TOBJECT'),'EMonitorActive
11831:   FindClass('TOBJECT'),'TIPCThread
11832:   TEventKind', '( evMonitorAttach, evMonitorDetach, evMonitorSigna
11833:   + 'l, evMonitorExit, evClientStart, evClientStop, evClientAttach, evClientDet
11834:   + 'ach, evClientSwitch, evClientSignal, evClientExit )
11835:   TClientFlag', '( cfError, cfMouseMove, cfMouseDown, cfResize, cfAttach )
11836:   TClientFlags', 'set of TClientFlag
11837:   //PEventData', '^TEventData // will not work
11838:   TEventData', 'record X : SmallInt; Y : SmallInt; Flag : TClientF
11839:   + 'lag; Flags : TClientFlags; end
11840:   TConnectEvent', 'Procedure ( Sender : TIPCThread; Connecting : Boolean)
11841:   TDirUpdateEvent', 'Procedure ( Sender : TIPCThread)
11842:   TIPCTotifyEvent', 'Procedure ( Sender : TIPCThread; Data : TEventData)
11843:   //PIPCEventInfo', '^TIPCEventInfo // will not work
11844:   TIPCEventInfo', 'record FID:Integer;FKind:TEventKind;FData:TEventData;end
11845:   SIRegister_TIPCEvent(CL);
11846:   //PClientDirRecords', '^TClientDirRecords // will not work
11847:   SIRegister_TClientDirectory(CL);
11848:   TIPCState', '( stInactive, stDisconnected, stConnected )
11849:   SIRegister_TIPCThread(CL);
11850:   SIRegister_TIPCMonitor(CL);
11851:   SIRegister_TIPCClient(CL);
11852:   Function IsMonitorRunning( var Hndl : THandle) : Boolean
11853: end;
11854:
11855: (*-----*)
11856: procedure SIRegister_ALGSMComm(CL: TPSPascalCompiler);
11857: begin
11858:   SIRegister_TALGSMComm(CL);
11859:   Function ALGSMComm_BuildPDUMessage( aSMSCenter, aSMSAddress, aMessage : AnsiString) : AnsiString
11860:   Procedure ALGSMComm_DecodePDUMessage(aPDUMessage:AnsiString;var aSMSCenter,aSMSAddress,
aMessage:AnsiString);
11861:   Function ALGSMComm_UnicodeToGSM7BitDefaultAlphabet( aMessage : WideString) : AnsiString
11862:   Function ALGSMComm_GSM7BitDefaultAlphabetToUnicode(aMess:AnsiString;const
UseGreekAlphabet:Bool):Widestring;
11863:   function ALMatchesMask(const Filename, Mask: AnsiString): Boolean;
11864: end;
11865:
11866: procedure SIRegister_ALHttpCommon(CL: TPSPascalCompiler);
11867: begin
11868:   TALHTTPPropertyChangeEvent', 'Procedure(sender:TObject;const PropertyIndex:Integer;
11869:   TALHTTPProtocolVersion', '( HTTPPv_1_0, HTTPPv_1_1 )
11870:   TALHTTPMethod', '(HTTPmt_Get,HTTPmt_Post,HTTPmt_Head,HTTPmt_Trace,HTTPmt_Put,HTTPmt_Delete);
11871:   TInternetScheme', 'integer
11872:   TALIPv6Binary', 'array[1..16] of Char;
11873:   // TALIPv6Binary = array[1..16] of ansiChar;
11874:   // TInternetScheme = Integer;
11875:   SIRegister_TALHTTPRequestHeader(CL);
11876:   SIRegister_TALHTTPCookie(CL);
11877:   SIRegister_TALHTTPCookieCollection(CL);
11878:   SIRegister_TALHTTPResponseHeader(CL);
11879:   Function ALHTTPDecode( const AStr : AnsiString) : AnsiString
11880:   Procedure ALHTTPEncodeParamNameValues( ParamValues : TALStrings)
11881:   // Procedure ALExtractHTTPFields(Separators, WhiteSpace, Quotes:TSysCharSet;
Content:PAnsiChar;Strings:TALStrings;StripQuotes:Boolean;
11882:   // Procedure ALExtractHeaderFields( Separators,WhiteSpace, Quotes : TSysCharSet; Content : PAnsiChar;
Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11883:   // Procedure ALExtractHeaderFieldsWithQuoteEscaped(Separators,WhiteSpace,
Quotes:TSysCharSet;Content:PAnsiChar;Strings : TALStrings; Decode : Boolean; StripQuotes : Boolean)
11884:   Function ALRemoveSchemeFromUrl( aUrl : AnsiString) : ansiString
11885:   Function ALExtractSchemeFromUrl( aUrl : AnsiString) : TInternetScheme
11886:   Function ALExtractHostNameFromUrl( aUrl : AnsiString) : AnsiString
11887:   Function ALExtractDomainNameFromUrl( aUrl : AnsiString) : AnsiString
11888:   Function ALExtractUriPathFromUrl( aUrl : AnsiString) : AnsiString

```

```

11889:  Function AlInternetCrackUrl( aUrl : AnsiString; var SchemeName,HostName,UserName,Password,UrlPath,
    ExtraInfo : AnsiString; var PortNumber : integer) : Boolean;
11890:  Function AlInternetCrackUrl1( aUrl : AnsiString; var SchemeName, HostName, UserName, Password,UrlPath,
    Anchor : AnsiString; Query : TALStrings; var PortNumber : integer) : Boolean;
11891:  Function AlInternetCrackUrl2(var Url:AnsiString;var Anchor:AnsiString;Query:TALStrings):Boolean;
11892:  Function AlRemoveAnchorFromUrl( aUrl : AnsiString; var aAnchor : AnsiString) : AnsiString;
11893:  Function AlRemoveAnchorFromUrl1( aUrl : AnsiString) : AnsiString;
11894:  Function AlCombineUrl( RelativeUrl, BaseUrl : AnsiString) : AnsiString;
11895:  Function AlCombineUrl1(RelativeUrl, BaseUrl, Anchor : AnsiString; Query:TALStrings) : AnsiString;
11896:  Function ALGmtDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
11897:  Function ALDateTimeToRfc822Str( const aValue : TDateTime) : AnsiString
11898:  Function ALTryRfc822StrToGMTDateTime( const S : AnsiString; out Value : TDateTime) : Boolean
11899:  Function ALRfc822StrToGMTDateTime( const s : AnsiString) : TDateTime
11900:  Function ALTryIPv4StrToNumeric( aIPv4Str : ansiString; var aIPv4Num : Cardinal) : Boolean
11901:  Function ALIPv4StrToNumeric( aIPv4 : ansiString) : Cardinal
11902:  Function ALNumericToIPv4Str( aIPv4 : Cardinal) : ansiString
11903:  Function ALZeroIPv6 : TALIPv6Binary
11904:  Function ALTryIPv6StrToBinary( aIPv6Str : ansiString; var aIPv6Bin : TALIPv6Binary) : Boolean
11905:  Function ALIPv6StrToBinary( aIPv6 : ansiString) : TALIPv6Binary
11906:  Function ALBinaryToIPv6Str( aIPv6 : TALIPv6Binary) : ansiString
11907:  Function ALBinaryStrToIPv6Binary( aIPv6BinaryStr : ansiString) : TALIPv6Binary
11908: end;
11909:
11910: procedure SIRegister_ALFcnHTML(CL: TPSPascalCompiler);
11911: begin
11912:   Procedure ALUTF8ExtractHTMLText(HtmlCont:AnsiString;LstExtractedResourceText:TALStrings;const
    DecodeHTMLText:Boolean);
11913:   Function ALUTF8ExtractHTMLText1(HtmlContent:AnsiString;const DecodeHTMLText:Boolean): AnsiString;
11914:   Function ALXMLCDATAElementEncode( Src : AnsiString) : AnsiString
11915:   Function ALXMLTextElementEncode(Src : AnsiString; const useNumericReference : boolean) : AnsiString
11916:   Function ALUTF8XMLTextElementDecode( const Src : AnsiString) : AnsiString
11917:   Function ALUTF8HTMLEncode(const Src:AnsiStr;const EncodeASCIHtmlEntities:Bool;const
    useNumRef:bool):AnsiString);
11918:   Function ALUTF8HTMLDecode( const Src : AnsiString) : AnsiString
11919:   Function ALJavaScriptEncode( const Src : AnsiString; const useNumericReference : boolean) : AnsiString
11920:   Function ALUTF8JavaScriptDecode( const Src : AnsiString) : AnsiString
11921:   Procedure ALHideHtmlUnwantedTagForHTMLHandleTagfunc(var HtmlContent:AnsiString; const
    DeleteBodyOfUnwantedTag : Boolean; const ReplaceUnwantedTagCharBy : AnsiChar)
11922:   Procedure ALCompactHtmlTagParams( TagParams : TALStrings)
11923: end;
11924:
11925: procedure SIRegister_ALInternetMessageCommon(CL: TPSPascalCompiler);
11926: begin
11927:   SIRegister_TAEmailHeader(CL);
11928:   SIRegister_TALNewsArticleHeader(CL);
11929:   Function ALParseEmailAddress(FriendlyEmail:AnsiString;var RealName:AString;const
    decodeRealName:Bool):AnsiString;
11930:   Function ALExtractEmailAddress( FriendlyEmail : AnsiString) : AnsiString
11931:   Function ALMakeFriendlyEmailAddress( aRealName, aEmail : AnsiString) : AnsiString
11932:   Function ALEncodeRealName4FriendlyEmailAddress( aRealName : AnsiString) : AnsiString
11933:   Function ALGenerateInternetMessageID : AnsiString;
11934:   Function ALGenerateInternetMessageID1( ahostname : AnsiString) : AnsiString;
11935:   Function ALDecodeQuotedPrintableString( src : AnsiString) : AnsiString
11936:   Function ALDecodeInternetMessageHeaderInUTF8(aHeaderStr:AnsiString;aDefaultCodePage:Integer):AnsiString;
11937: end;
11938:
11939: (*-----*)
11940: procedure SIRegister_ALFcnWinSock(CL: TPSPascalCompiler);
11941: begin
11942:   Function ALHostToIP( HostName : AnsiString; var Ip : AnsiString):Boolean
11943:   Function ALIPAddrToName( IPAddr : AnsiString) : AnsiString
11944:   Function ALgetLocalIPs : TALStrings
11945:   Function ALgetLocalHostName : AnsiString
11946: end;
11947:
11948: procedure SIRegister_ALFcnCGI(CL: TPSPascalCompiler);
11949: begin
11950:   Procedure ALCGIInitDefaultServerVariablesFromWebRequest(WebRequest :
    TALWebRequest;ServerVariables:TALStrings);
11951:   Procedure ALCGIInitDefaultServerVariablesFromWebRequest1(WebRequest: TALWebRequest; ServerVariables :
    TALStrings; ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
11952:   Procedure ALCGIInitDefaultServerVariables( ServerVariables : TALStrings);
11953:   Procedure ALCGIInitDefaultServerVariables1(ServerVars:TALStrings;ScriptName,
    ScriptFileName:AnsiString;Url:AnsiString);
11954:   Procedure ALCGIInitServerVariablesFromWebRequest( WebRequest:TALWebRequest; ServerVariables : TALStrings;
    ScriptName, ScriptFileName : AnsiString; Url : AnsiString);
11955:   Procedure ALCGIExec( InterpreterFilename : AnsiString; ServerVariables : TALStrings; RequestContentStream
    : Tstream; ResponseContentStream : Tstream; ResponseHeader : TALHTTPResponseHeader);
11956:   Procedure ALCGIExec1(ScriptName,ScriptFileName, Url, X_REWRITE_URL, InterpreterFilename:AnsiString;
    WebRequest : TALIsapiRequest;
    overloadedCookies:AnsiString;overloadedQueryString:AnsiString;overloadedReferer: AnsiString;'
    + 'overloadedRequestContentStream:Tstream;var
    ResponseContentStr:AnsiString;ResponseHeader:TALHTTPResponseHeader;
11957:   Procedure ALCGIExec2(ScriptName,ScriptFileName,Url,X_REWRITE_URL,
    InterpreterFilename:AnsiString;WebRequest: TALIsapiRequest; var ResponseContentString : AnsiString;
    ResponseHeader : TALHTTPResponseHeader);
11958: end;
11959:
11960:
11961: procedure SIRegister_ALFcnExecute(CL: TPSPascalCompiler);

```

```

11962: begin
11963:   TStartupInfoA, 'TStartupInfo
11964:   'SE_CREATE_TOKEN_NAME', 'String').SetString( 'SeCreateTokenPrivilege
11965:   SE_ASSIGNPRIMARYTOKEN_NAME', 'String').SetString( 'SeAssignPrimaryTokenPrivilege
11966:   SE_LOCK_MEMORY_NAME', 'String').SetString( 'SeLockMemoryPrivilege
11967:   SE_INCREASE_QUOTA_NAME', 'String').SetString( 'SeIncreaseQuotaPrivilege
11968:   SE_UNSOLICITED_INPUT_NAME', 'String').SetString( 'SeUnsolicitedInputPrivilege
11969:   SE_MACHINE_ACCOUNT_NAME', 'String').SetString( 'SeMachineAccountPrivilege
11970:   SE_TCB_NAME', 'String').SetString( 'SeTcbPrivilege
11971:   SE_SECURITY_NAME', 'String').SetString( 'SeSecurityPrivilege
11972:   SE_TAKE_OWNERSHIP_NAME', 'String').SetString( 'SeTakeOwnershipPrivilege
11973:   SE_LOAD_DRIVER_NAME', 'String').SetString( 'SeLoadDriverPrivilege
11974:   SE_SYSTEM_PROFILE_NAME', 'String').SetString( 'SeSystemProfilePrivilege
11975:   SE_SYSTEMTIME_NAME', 'String').SetString( 'SeSystemtimePrivilege
11976:   SE_PROF_SINGLE_PROCESS_NAME', 'String').SetString( 'SeProfileSingleProcessPrivilege
11977:   SE_INC_BASE_PRIORITY_NAME', 'String').SetString( 'SeIncreaseBasePriorityPrivilege
11978:   SE_CREATE_PAGEFILE_NAME', 'String').SetString( 'SeCreatePagefilePrivilege
11979:   SE_CREATE_PERMANENT_NAME', 'String').SetString( 'SeCreatePermanentPrivilege
11980:   SE_BACKUP_NAME', 'String').SetString( 'SeBackupPrivilege
11981:   SE_RESTORE_NAME', 'String').SetString( 'SeRestorePrivilege
11982:   SE_SHUTDOWN_NAME', 'String').SetString( 'SeShutdownPrivilege
11983:   SE_DEBUG_NAME', 'String').SetString( 'SeDebugPrivilege
11984:   SE_AUDIT_NAME', 'String').SetString( 'SeAuditPrivilege
11985:   SE_SYSTEM_ENVIRONMENT_NAME', 'String').SetString( 'SeSystemEnvironmentPrivilege
11986:   SE_CHANGE_NOTIFY_NAME', 'String').SetString( 'SeChangeNotifyPrivilege
11987:   SE_REMOTE_SHUTDOWN_NAME', 'String').SetString( 'SeRemoteShutdownPrivilege
11988:   SE_UNDOCK_NAME', 'String').SetString( 'SeUndockPrivilege
11989:   SE_SYNC_AGENT_NAME', 'String').SetString( 'SeSyncAgentPrivilege
11990:   SE_ENABLE_DELEGATION_NAME', 'String').SetString( 'SeEnableDelegationPrivilege
11991:   SE_MANAGE_VOLUME_NAME', 'String').SetString( 'SeManageVolumePrivilege
11992:   Function ALGetEnvironmentString : AnsiString
11993:   Function ALWinExec32(const FileName, CurrentDir,
Environment: AnsiString; InStream: Tstream; OutStream: Tstream): Dword;
11994:   Function ALWinExec321(const FileName: AnsiString; InputStream: Tstream; OutputStream: Tstream): Dword;
11995:   Function ALWinExecAndWait32( FileName : AnsiString; Visibility : integer ) : DWORD
11996:   Function ALWinExecAndWait32V2( FileName : AnsiString; Visibility : integer ) : DWORD
11997:   Function ALNTSetPrivilege( sPrivilege : AnsiString; bEnabled : Boolean ) : Boolean
11998: end;
11999:
12000: procedure SIRegister_ALFcnFile(CL: TPSPascalCompiler);
12001: begin
12002:   Function ALEmptyDirectory(Directory: ansiString; SubDirectory: Bool; IgnoreFiles: array of AnsiString; const
RemoveEmptySubDirectory : Boolean; const FileNameMask : ansiString; const MinFileAge : TdateTime): Boolean;
12003:   Function ALEmptyDirectory1( Directory : ansiString; SubDirectory : Boolean; const
RemoveEmptySubDirectory: Bool; const FileNameMask : ansiString; const MinFileAge : TdateTime ) : Boolean;
12004:   Function ALCopyDirectory( SrcDirectory, DestDirectory : ansiString; SubDirectory : Boolean; const
FileNameMask : ansiString; const FailIfExists : Boolean ) : Boolean
12005:   Function ALGetModuleName : ansistring
12006:   Function ALGetModuleFileNameWithoutExtension : ansistring
12007:   Function ALGetModulePath : ansiString
12008:   Function ALGetFileSize( const AFileName : ansistring ) : int64
12009:   Function ALGetFileVersion( const AFileName : ansistring ) : ansiString
12010:   Function ALGetFileCreationDateTime( const aFileName : AnsiString ) : TdateTime
12011:   Function ALGetFileLastWriteDateTime( const aFileName : AnsiString ) : TdateTime
12012:   Function ALGetFileLastAccessDateTime( const aFileName : AnsiString ) : TdateTime
12013:   Procedure ALSetFileCreationDateTime( const aFileName : AnsiString; const aCreationDateTime : TdateTime)
12014:   Function ALIsDirectoryEmpty( const directory : ansiString ) : boolean
12015:   Function ALFileExists( const Path : ansiString ) : boolean
12016:   Function ALDirectoryExists( const Directory : AnsiString ) : Boolean
12017:   Function ALCreateDir( const Dir : AnsiString ) : Boolean
12018:   Function ALRemoveDir( const Dir : AnsiString ) : Boolean
12019:   Function ALDeleteFile( const FileName : AnsiString ) : Boolean
12020:   Function ALRenameFile( const OldName, NewName : ansistring ) : Boolean
12021: end;
12022:
12023: procedure SIRegister_ALFcnMime(CL: TPSPascalCompiler);
12024: begin
12025:   NativeInt', 'Integer
12026:   NativeUInt', 'Cardinal
12027:   Function ALMimeBase64EncodeString( const S : AnsiString ) : AnsiString
12028:   Function ALMimeBase64EncodeStringNoCRLF( const S : AnsiString ) : AnsiString
12029:   Function ALMimeBase64DecodeString( const S : AnsiString ) : AnsiString
12030:   Function ALMimeBase64EncodedSize( const InputSize : NativeInt ) : NativeInt
12031:   Function ALMimeBase64EncodedSizeNoCRLF( const InputSize : NativeInt ) : NativeInt
12032:   Function ALMimeBase64DecodedSize( const InputSize : NativeInt ) : NativeInt
12033:   Procedure ALMimeBase64Encode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12034:   Procedure ALMimeBase64EncodeNoCRLF( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12035:   Procedure ALMimeBase64EncodeFullLines( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt)
12036:   Function ALMimeBase64Decode( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt ) : NativeInt;
12037:   Function ALMimeBase64DecodePartial( const InputBuffer : TByteDynArray; InputOffset : NativeInt; const
InputByteCount : NativeInt; out OutputBuffer : TByteDynArray; OutputOffset : NativeInt;
+ 'var ByteBuffer : Cardinal; var ByteBufferSpace : Cardinal ) : NativeInt;
12038:   Function ALMimeBase64DecodePartialEnd( out OutputBuffer : TByteDynArray; OutputOffset : NativeInt; const
ByteBuffer : Cardinal; const ByteBufferSpace : Cardinal ) : NativeInt;
12039:   Procedure ALMimeBase64Encode(const InputBuf: TByteDynArray; const InputByteCnt: NatInt; out
OutputBuf: TByteDynArray);

```

```

12041: Procedure ALMimeBase64EncodeNoCRLF(const InputBuffer:TByteDynArray; const InputByteCount:NativeInt;out
OutputBuffer:TByteDynArray);
12042: Procedure ALMimeBase64EncodeFullLines(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
OutputBuffer:TByteDynArray);
12043: Function ALMimeBase64Decode1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
OutputBuffer:TByteDynArray):NativeInt;
12044: Function ALMimeBase64DecodePartial1(const InputBuffer:TByteDynArray;const InputByteCount:NativeInt;out
OutputBuffer: TByteDynArray; var ByteBuffer: Cardinal; var ByteBufferSpace: Cardinal): NativeInt;
12045: Function ALMimeBase64DecodePartialEnd1(out OutputBuffer:TByteDynArray;const ByteBuffer:Cardinal;const
ByteBufferSpace:Cardinal):NativeInt;
12046: Procedure ALMimeBase64EncodeFile(const InputFileName, OutputFileName: TFileName)
12047: Procedure ALMimeBase64EncodeFileNoCRLF(const InputFileName, OutputFileName: TFileName)
12048: Procedure ALMimeBase64DecodeFile(const InputFileName, OutputFileName: TFileName)
12049: Procedure ALMimeBase64EncodeStream(const InputStream: TStream; const OutputStream: TStream)
12050: Procedure ALMimeBase64EncodeStreamNoCRLF(const InputStream: TStream; const OutputStream: TStream)
12051: Procedure ALMimeBase64DecodeStream(const InputStream: TStream; const OutputStream: TStream)
12052: 'cALMimeBase64_ENCODED_LINE_BREAK', 'LongInt').SetInt( 76);
12053: 'cALMimeBase64_DECODED_LINE_BREAK', 'LongInt').SetInt( cALMimeBase64_ENCODED_LINE_BREAK div 4 * 3);
12054: 'cALMimeBase64_BUFFER_SIZE', 'LongInt').SetInt( cALMimeBase64_DECODED_LINE_BREAK * 3 * 4 * 4);
12055: Procedure ALFillMimeContentTypeByExtList( AMIMEList: TALStrings)
12056: Procedure ALFillExtByMimeContentTypeList( AMIMEList: TALStrings)
12057: Function ALGetDefaultFileExtFromMimeContentType( aContentType: AnsiString): AnsiString
12058: Function ALGetDefaultMIMEContentTypeFromExt( aExt: AnsiString): AnsiString
12059: end;
12060:
12061: procedure SIRegister_ALXmlDoc(CL: TPSPascalCompiler);
12062: begin
12063: 'cALXMLNodeMaxListSize', 'LongInt').SetInt( Maxint div 16);
12064: FindClass('TOBJECT'), 'TALXMLNode
12065: FindClass('TOBJECT'), 'TALXMLNodeList
12066: FindClass('TOBJECT'), 'TALXMLDocument
12067: TALXMLParseProcessingInstructionEvent', Procedure (Sender:TObject; const Target,Data:AnsiString)
12068: TALXMLParseTextEvent', Procedure ( Sender: TObject; const str: AnsiString)
12069: TALXMLParseStartElementEvent', Procedure ( Sender: TObject; co
12070: + 'nt Name: AnsiString; const Attributes: TALStrings)
12071: TALXMLParseEndElementEvent', Procedure ( Sender: TObject; const Name: AnsiString)
12072: TALXMLNodeType', '( ntReserved, ntElement, ntAttribute, ntText, '
12073: + 'ntCDATA, ntEntityRef, ntEntity, ntProcessingInstr, ntComment, ntDocument, '
12074: + 'ntDocType, ntDocFragment, ntNotation )
12075: TALXMLDocOption', '( doNodeAutoCreate, doNodeAutoIndent )
12076: TALXMLDocOptions', set of TALXMLDocOption
12077: TALXMLParseOption', '( poPreserveWhiteSpace, poIgnoreXMLReferences )
12078: TALXMLParseOptions', set of TALXMLParseOption
12079: TALXMLPrologItem', '( xpVersion, xpEncoding, xpStandalone )
12080: PALPointerXMLNodeList', '^TALPointerXMLNodeList // will not work
12081: SIRegister_EALXMLDocError(CL);
12082: SIRegister_TALXMLNodeList(CL);
12083: SIRegister_TALXMLNode(CL);
12084: SIRegister_TALXmlElementNode(CL);
12085: SIRegister_TALXmlAttributeNode(CL);
12086: SIRegister_TALXmlTextNode(CL);
12087: SIRegister_TALXmlDocumentNode(CL);
12088: SIRegister_TALXmlCommentNode(CL);
12089: SIRegister_TALXmlProcessingInstrNode(CL);
12090: SIRegister_TALXmlCDATADataNode(CL);
12091: SIRegister_TALXmlEntityRefNode(CL);
12092: SIRegister_TALXmlEntityNode(CL);
12093: SIRegister_TALXmlDocTypeNode(CL);
12094: SIRegister_TALXmlDocFragmentNode(CL);
12095: SIRegister_TALXmlNotationNode(CL);
12096: SIRegister_TALXMLDocument(CL);
12097: cALXMLUTF8EncodingStr', 'String').SetString( 'UTF-8
12098: cALXMLUTF8HeaderStr', 'String').SetString('<?xmlversion="1.0"encoding="UTF-8"standalone="yes"?'+#13#10);
12099: CALNSDelim', 'String').SetString( '
12100: CALXML', 'String').SetString( 'xml
12101: CALVersion', 'String').SetString( 'version
12102: CALEncoding', 'String').SetString( 'encoding
12103: CALStandalone', 'String').SetString( 'standalone
12104: CALDefaultNodeIndent', 'String').SetString( '
12105: CALXmlDocument', 'String').SetString( 'DOCUMENT
12106: Function ALCreateEmptyXMLDocument( const Rootname: AnsiString): TalXMLDocument
12107: Procedure ALClearXMLDocument(const rootname:AnsiString;xmlDoc:TalXMLDocument;const
EncodingStr:AnsiString);
12108: Function ALFindXmlNodeByChildNodeValue(xmlrec: TalxmlNode; const ChildNodeName,
ChildNodeValue: AnsiString; const Recurse: Boolean): TalxmlNode
12109: Function ALFindXmlNodeByNameAndChildNodeValue( xmlrec: TalxmlNode; const NodeName: ansiString; const
ChildNodeName, ChildNodeValue: AnsiString; const Recurse: Boolean): TalxmlNode
12110: Function ALFindXmlNodeByAttribute(xmlrec: TalxmlNode; const AttributeName, AttributeValue: AnsiString; const
Recurse: Boolean): TalxmlNode
12111: Function ALFindXmlNodeByNameAndAttribute( xmlrec: TalxmlNode; const NodeName: ansiString; const
AttributeName, AttributeValue: AnsiString; const Recurse: Boolean): TalxmlNode
12112: Function ALExtractAttrValue( const AttrName, AttrLine: AnsiString; const Default: AnsiString):
AnsiString
12113: end;
12114:
12115: procedure SIRegister_TeCanvas(CL: TPSPascalCompiler);
12116: //based on TEEProc, TeCanvas, TEEngine, TChart
12117: begin
12118: 'TeePiStep', 'Double').setExtended( Pi / 180.0);

```



```

12119: 'TeeDefaultPerspective','LongInt').SetInt( 100);
12120: 'TeeMinAngle','LongInt').SetInt( 270);
12121: 'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12122: 'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12123: 'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ));
12124: 'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12125: 'teeclMoneyGreen','LongWord').SetUInt( TColor ( $C0DCC0 ));
12126: 'teeclSkyBlue','LongWord').SetUInt( TColor ( $F0CAA6 ));
12127: 'teeclCream','LongWord').SetUInt( TColor ( $F0FBFF ));
12128: 'teeclMedGray','LongWord').SetUInt( TColor ( $A4A0A0 ));
12129: 'TA_LEFT','LongInt').SetInt( 0);
12130: 'TA_RIGHT','LongInt').SetInt( 2);
12131: 'TA_CENTER','LongInt').SetInt( 6);
12132: 'TA_TOP','LongInt').SetInt( 0);
12133: 'TA_BOTTOM','LongInt').SetInt( 8);
12134: 'teePATCOPY','LongInt').SetInt( 0);
12135: 'NumCirclePoints','LongInt').SetInt( 64);
12136: 'teeDEFAULT_CHARSET','LongInt').SetInt( 1);
12137: 'teeANTIALIASED_QUALITY','LongInt').SetInt( 4);
12138: 'TA_LEFT','LongInt').SetInt( 0);
12139: 'bs_Solid','LongInt').SetInt( 0);
12140: 'teepf24Bit','LongInt').SetInt( 0);
12141: 'teepfDevice','LongInt').SetInt( 1);
12142: 'CM_MOUSELEAVE','LongInt').SetInt( 10000);
12143: 'CM_SYSCOLORCHANGE','LongInt').SetInt( 10001);
12144: 'DC_BRUSH','LongInt').SetInt( 18);
12145: 'DC_PEN','LongInt').SetInt( 19);
12146: teeCOLORREF, 'LongWord
12147: TLogBrush', 'record lbStyle : Integer; lbColor : TColor; lbHatch: Integer; end
12148: //TNotifyEvent', 'Procedure ( Sender : TObject)
12149: SIRegister_TFilterRegion(CL);
12150: SIRegister_IFormCreator(CL);
12151: SIRegister_TTeeFilter(CL);
12152: //TFilterClass', 'class of TTeeFilter
12153: SIRegister_TFilterItems(CL);
12154: SIRegister_TConvolveFilter(CL);
12155: SIRegister_TBlurFilter(CL);
12156: SIRegister_TTeePicture(CL);
12157: TPenEndStyle', '( esRound, esSquare, esFlat )
12158: SIRegister_TChartPen(CL);
12159: SIRegister_TChartHiddenPen(CL);
12160: SIRegister_TDottedGrayPen(CL);
12161: SIRegister_TDarkGrayPen(CL);
12162: SIRegister_TWhitePen(CL);
12163: SIRegister_TChartBrush(CL);
12164: TTeeView3DScrolled', 'Procedure ( IsHoriz : Boolean)
12165: TTeeView3DChangedZoom', 'Procedure ( NewZoom : Integer)
12166: SIRegister_TView3DOptions(CL);
12167: FindClass('TOBJECT'),'TTeeCanvas
12168: TTeeTransparency', 'Integer
12169: SIRegister_TTeeBlend(CL);
12170: FindClass('TOBJECT'),'TCanvas3D
12171: SIRegister_TTeeShadow(CL);
12172: teeTGradientDirection', '( gdTopBottom, gdBottomTop, gdLeftRight, g'
12173: + 'dRightLeft, gdFromCenter, gdFromTopLeft, gdFromBottomLeft, gdRadial,gdDiagonalUp,gdDiagonalDown )
12174: FindClass('TOBJECT'),'TSubGradient
12175: SIRegister_TCustomTeeGradient(CL);
12176: SIRegister_TSubGradient(CL);
12177: SIRegister_TTeeGradient(CL);
12178: SIRegister_TTeeFontGradient(CL);
12179: SIRegister_TTeeFont(CL);
12180: TCanvasBackMode', '( cbmNone, cbmTransparent, cbmOpaque )
12181: TCanvasTextAlign', 'Integer
12182: TTeeCanvasHandle', 'HDC
12183: SIRegister_TTeeCanvas(CL);
12184: TPoint3DFloat', 'record X : Double; Y : Double; Z : Double; end
12185: SIRegister_TFloatXYZ(CL);
12186: TPoint3D', 'record x : integer; y : integer; z : Integer; end
12187: TRGB', 'record blue: byte; green: byte; red: byte; end
12188: {TRGB=packed record
12189:   Blue : Byte;
12190:   Green : Byte;
12191:   Red : Byte;
12192:   //$IFDEF CLX //Alpha : Byte; // Linux end;}
12193:
12194: TTeeCanvasCalcPoints', 'Function ( x, z : Integer; var P0, P1 : '
12195:   + 'TPoint3D; var Color0, Color1 : TColor) : Boolean
12196: TTeeCanvasSurfaceStyle', '( tcsSolid, tcsWire, tcsDot )
12197: TCanvas3DPlane', '( cpX, cpY, cpZ )
12198: //IInterface', 'IUnknown
12199: SIRegister_TCanvas3D(CL);
12200: SIRegister_TTeeCanvas3D(CL);
12201: TTrianglePoints', 'Array[0..2] of TPoint;
12202: TFourPoints', 'Array[0..3] of TPoint;
12203: Function ApplyDark( Color : TColor; HowMuch : Byte) : TColor
12204: Function ApplyBright( Color : TColor; HowMuch : Byte) : TColor
12205: Function Point3D( const x, y, z : Integer) : TPoint3D
12206: Procedure SwapDouble( var a, b : Double)
12207: Procedure SwapInteger( var a, b : Integer)

```

```

12208: Procedure RectSize( const R : TRect; var RectWidth, RectHeight : Integer)
12209: Procedure teeRectCenter( const R : TRect; var X, Y : Integer)
12210: Function RectFromPolygon( const Points : array of TPoint; NumPoints : Integer): TRect
12211: Function RectFromTriangle( const Points : TTrianglePoints) : TRect
12212: Function RectangleInRectangle( const Small, Big : TRect) : Boolean
12213: Procedure ClipCanvas( ACanvas : TCanvas; const Rect : TRect)
12214: Procedure UnClipCanvas( ACanvas : TCanvas)
12215: Procedure ClipEllipse( ACanvas : TCanvas; const Rect : TRect)
12216: Procedure ClipRoundRectangle(ACanvas:TCanvas;const Rect : TRect; RoundSize : Integer)
12217: Procedure ClipPolygon(ACanvas:TCanvas;const Points:array of TPoint;NumPoints:Integer)
12218: 'TeeCharForHeight','String').SetString( 'W
12219: 'DarkerColorQuantity','Byte').SetUInt( 128);
12220: 'DarkColorQuantity','Byte').SetUInt( 64);
12221: TButtonGetColorProc', 'Function : TColor
12222: SIRegister_TTeeButton(CL);
12223: SIRegister_TButtonColor(CL);
12224: SIRegister_TComboFlat(CL);
12225: Procedure TeeSetTeePen(FPen:TPen; APen : TChartPen; AColor : TColor; Handle:TCanvasHandle)
12226: Function TeePoint( const aX, aY : Integer) : TPoint
12227: Function TEETestPointInRect( const Rect : TRect; const P : TPoint) : Boolean;
12228: Function PointInRect1( const Rect : TRect; x, y : Integer) : Boolean;
12229: Function TeeRect( Left, Top, Right, Bottom : Integer) : TRect
12230: Function OrientRectangle( const R : TRect) : TRect
12231: Procedure TeeSetBitmapSize( Bitmap : TBitmap; Width, Height : Integer)
12232: Function PolygonBounds( const P : array of TPoint) : TRect
12233: Function PolygonInPolygon( const A, B : array of TPoint) : Boolean
12234: Function RGBValue( const Color : TColor) : TRGB
12235: Function EditColor( AOwner : TComponent; AColor : TColor) : TColor
12236: Function EditColorDialog( AOwner : TComponent; var AColor : TColor) : Boolean
12237: Function PointAtDistance( AFrom, ATo : TPoint; ADist : Integer) : TPoint
12238: Function TeeCull( const P : TFourPoints) : Boolean;
12239: Function TeeCull1( const P0, P1, P2 : TPoint) : Boolean;
12240: TSmoothStretchOption', '( ssBestQuality, ssBestPerformance )
12241: Procedure SmoothStretch( Src, Dst : TBitmap);
12242: Procedure SmoothStretch1( Src, Dst : TBitmap; Option : TSmoothStretchOption);
12243: Function TeeDistance( const x, y : Double) : Double
12244: Function TeeLoadLibrary( const FileName : String) : HInst
12245: Procedure TeeFreeLibrary( hLibModule : HMODULE)
12246: Procedure TeeBlendBitmaps( const Percent : Double; ABitmap, BBitmap : TBitmap; BOrigin : TPoint)
12247: //Procedure TeeCalcLines( var Lines : TRGBArray; Bitmap : TBitmap)
12248: Procedure TeeShadowSmooth(Bitmap, Back : TBitmap; Left, Top, Width, Height, horz, vert : Integer;
Smoothness : Double; FullDraw : Boolean; ACanvas : TCanvas3D; Clip : Boolean)
12249: SIRegister_ICanvasHyperlinks(CL);
12250: SIRegister_ICanvasToolTips(CL);
12251: Function Supports( const Instance : IInterface; const IID : TGUID) : Boolean
12252: end;
12253:
12254: procedure SIRegister_ovcmisc(CL: TPSPascalCompiler);
12255: begin
12256:   TOvcHdc, 'Integer
12257:   TOvcHWND, 'Cardinal
12258:   TOvcHdc, 'HDC
12259:   TOvcHWND, 'HWND
12260: Function LoadBaseBitmap( lpBitmapName : PChar) : HBITMAP
12261: Function LoadBaseCursor( lpCursorName : PChar) : HCURSOR
12262: Function ovCompStruct( const S1, S2, Size : Cardinal) : Integer
12263: Function DefaultEpoch : Integer
12264: Function DrawButtonFrame(Canvas:TCanvas;const Client:TRect;IsDown,IsFlat:Bool;Style:TButtonStyle):TRect;
12265: Procedure FixRealPrim( P : PChar; DC : Char)
12266: Function GetDisplayString( Canvas : TCanvas; const S : string; MinChars, MaxWidth : Integer) : string
12267: Function GetLeftButton : Byte
12268: Function GetNextDlgItem( Ctrl : TOvcHwnd) : hWnd
12269: Procedure GetRGB( Clr : TColor; var IR, IG, IB : Byte)
12270: Function GetShiftFlags : Byte
12271: Function ovCreateRotatedFont( F : TFont; Angle : Integer) : hFont
12272: Function GetTopTextMargin(Font:TFont;BorderStyle:TBorderStyle; Height:Integer;Ctl3D:Boolean): Integer
12273: Function ovExtractWord( N : Integer; const S : string; WordDelims : TCharSet) : string
12274: Function ovIsForegroundTask : Boolean
12275: Function ovTrimLeft( const S : string) : string
12276: Function ovTrimRight( const S : string) : string
12277: Function ovQuotedStr( const S : string) : string
12278: Function ovWordCount( const S : string; const WordDelims : TCharSet) : Integer
12279: Function ovWordPosition(const N:Integer;const S: string;const WordDelims : TCharSet) : Integer
12280: Function PtrDiff( const P1, P2 : PChar) : Word
12281: Procedure PtrInc( var P, Delta : Word)
12282: Procedure PtrDec( var P, Delta : Word)
12283: Procedure FixTextBuffer( InBuf, OutBuf : PChar; OutSize : Integer)
12284: Procedure TransStretchBlt( DstDC : TOvcHdc; DstX, DstY, DstW, DstH : Integer; SrcDC : TOvcHdc; SrcX, SrcY,
SrcW, SrcH : Integer; MaskDC : TOvcHdc; MaskX, MaskY : Integer)
12285: Function ovMinI( X, Y : Integer) : Integer
12286: Function ovMaxI( X, Y : Integer) : Integer
12287: Function ovMinL( X, Y : LongInt) : LongInt
12288: Function ovMaxL( X, Y : LongInt) : LongInt
12289: Function GenerateComponentName( PF : TWinControl; const Root : string) : string
12290: Function PartialCompare( const S1, S2 : string) : Boolean
12291: Function PathEllipsis( const S : string; MaxWidth : Integer) : string
12292: Function ovCreateDisabledBitmap( FOriginal : TBitmap; OutlineColor : TColor) : TBitmap
12293: Procedure ovCopyParentImage( Control : TControl; Dest : TCanvas)
12294: Procedure ovDrawTransparentBitmap( Dest : TCanvas; X, Y, W, H : Integer; Rect : TRect; Bitmap : TBitmap;
TransparentColor : TColor)

```

```

12295: Procedure DrawTransparentBitmapPrim(DC:TovcHdc;Bitmap:HBitmap;xStart,yStart,Width,Height:Integer;Rect:TRect;
TransparentColor : TColorRef)
12296: Function ovWidthOf( const R : TRect ) : Integer
12297: Function ovHeightOf( const R : TRect ) : Integer
12298: Procedure ovDebugOutput( const S : string)
12299: Function GetArrowWidth( Width, Height : Integer ) : Integer
12300: Procedure StripCharSeq( CharSeq : string; var Str : string)
12301: Procedure StripCharFromEnd( aChr : Char; var Str : string)
12302: Procedure StripCharFromFront( aChr : Char; var Str : string)
12303: Function SystemParametersInfo( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT ) : BOOL
12304: Function SystemParametersInfoNC(uiAction,uiParam:UINT;pvParam:TNonClientMetrics;fWinIni:UINT):BOOL;
12305: Function SystemParametersInfoA( uiAction, uiParam : UINT; pvParam : UINT; fWinIni : UINT ) : BOOL
12306: Function CreateEllipticRgn( p1, p2, p3, p4 : Integer ) : HRGN
12307: Function CreateEllipticRgnIndirect( const p1 : TRect ) : HRGN
12308: Function CreateFontIndirect( const p1 : TLogFont ) : HFONT
12309: Function CreateMetaFile( p1 : PChar ) : HDC
12310: Function DescribePixelFormat(DC: HDC;p2:Integer;p3:UINT;var p4:TPixelFormatDescriptor): BOOL
12311: Function DrawText(hDC:HDC;lpString:PChar;nCount:Integer; var lpRect : TRect;uFormat:UINT):Integer
12312: Function DrawTextS(hDC:HDC;lpString:string;nCount:Integer; var lpRect:TRect;uFormat:UINT):Integer
12313: Function SetMapperFlags( DC : HDC; Flag : DWORD) : DWORD
12314: Function SetGraphicsMode( hdc : HDC; iMode : Integer ) : Integer
12315: Function SetMapMode( DC : HDC; p2 : Integer ) : Integer
12316: Function SetMetaFileBitsEx( Size : UINT; const Data : PChar ) : HMETAFILE
12317: //Function SetPaletteEntries(Palette:HPALETTE;StartIndex,NumEntries:UINT;var PaletteEntries):UINT
12318: Function SetPixel( DC : HDC; X, Y : Integer; Color : COLORREF) : COLORREF
12319: Function SetPixelV( DC : HDC; X, Y : Integer; Color : COLORREF) : BOOL
12320: //Function SetPixelFormat( DC : HDC; PixelFormat : Integer; FormatDef : PPixelFormatDescriptor ) : BOOL
12321: Function SetPolyFillMode( DC : HDC; PolyFillMode : Integer ) : Integer
12322: Function StretchBlt(DestDC:HDC;X,Y,Width,Height:Integer;SrcDC:HDC;XSrc,YSrc,SrcWidth,
SrcHeight:Integer;Rop:DWORD):BOOL
12323: Function SetRectRgn( Rgn : HRGN; X1, Y1, X2, Y2 : Integer ) : BOOL
12324: Function StretchDIBits(DC : HDC; DestX, DestY, DestWidth, DestHeight, SrcX, SrcY, SrcWidth,
SrcHeight:Integer;Bits:Integer; var BitsInfo : TBitmapInfo; Usage : UINT; Rop : DWORD) : Integer
12325: Function SetROP2( DC : HDC; p2 : Integer ) : Integer
12326: Function SetStretchBltMode( DC : HDC; StretchMode : Integer ) : Integer
12327: Function SetSystemPaletteUse( DC : HDC; p2 : UINT) : UINT
12328: Function SetTextCharacterExtra( DC : HDC; CharExtra : Integer ) : Integer
12329: Function SetTextColor( DC : HDC; Color : COLORREF) : COLORREF
12330: Function SetTextAlign( DC : HDC; Flags : UINT) : UINT
12331: Function SetTextJustification( DC : HDC; BreakExtra, BreakCount : Integer ) : Integer
12332: Function UpdateColors( DC : HDC ) : BOOL
12333: Function GetViewportExtEx( DC : HDC; var Size : TSize ) : BOOL
12334: Function GetViewportOrgEx( DC : HDC; var Point : TPoint ) : BOOL
12335: Function GetWindowExtEx( DC : HDC; var Size : TSize ) : BOOL
12336: Function GetWindowOrgEx( DC : HDC; var Point : TPoint ) : BOOL
12337: Function IntersectClipRect( DC : HDC; X1, Y1, X2, Y2 : Integer ) : Integer
12338: Function InvertRgn( DC : HDC; p2 : HRGN ) : BOOL
12339: Function MaskBlt( DestDC : HDC; XDest, YDest, Width, Height : Integer; SrcDC : HDC; XSrc, YSrc : Integer;
Mask : HBITMAP; xMask, yMask : Integer; Rop : DWORD) : BOOL
12340: Function PlgBlt(DestDC:HDC;const PtsArray,SrcDC:HDC;XSrc,YSrc,Widt,Heigh:Integer;Mask:HBITMAP;xMask,
yMask:Integer):BOOL;
12341: Function OffsetClipRgn( DC : HDC; XOffset, YOffset : Integer ) : Integer
12342: Function OffsetRgn( RGN : HRGN; XOffset, YOffset : Integer ) : Integer
12343: Function PatBlt( DC : HDC; X, Y, Width, Height : Integer; Rop : DWORD) : BOOL
12344: Function Pie( DC : HDC; X1, Y1, X2, Y2, X3, Y3, X4, Y4 : Integer ) : BOOL
12345: Function PlayMetaFile( DC : HDC; MF : HMETAFILE) : BOOL
12346: Function PaintRgn( DC : HDC; RGN : HRGN ) : BOOL
12347: Function PtInRegion( RGN : HRGN; X, Y : Integer ) : BOOL
12348: Function PtVisible( DC : HDC; X, Y : Integer ) : BOOL
12349: Function RectInRegion( RGN : HRGN; const Rect : TRect ) : BOOL
12350: Function RectVisible( DC : HDC; const Rect : TRect ) : BOOL
12351: Function Rectangle( DC : HDC; X1, Y1, X2, Y2 : Integer ) : BOOL
12352: Function RestoreDC( DC : HDC; SavedDC : Integer ) : BOOL
12353: end;
12354:
12355: procedure SIRegister_ovcfile(CL: TPSPascalCompiler);
12356: begin
12357: SIRegister_TovcAbstractStore(CL);
12358: //PEXPropInfo', '^TexPropInfo // will not work
12359: // TexPropInfo', 'record PI : TPropInfo; AObject : TObject; end
12360: SIRegister_TovcPropertyList(CL);
12361: SIRegister_TovcDataFiler(CL);
12362: Procedure UpdateStoredList( AForm : TWinControl; AStoredList : TStrings; FromForm : Boolean)
12363: Procedure UpdateStoredList1( AForm : TCustomForm; AStoredList : TStrings; FromForm : Boolean)
12364: Function CreateStoredItem( const CompName, PropName : string) : string
12365: Function ParseStoredItem( const Item : string; var CompName, PropName : string) : Boolean
12366: //Function GetPropType( PropInfo : PEXPropInfo ) : PTypeInfo
12367: end;
12368:
12369: procedure SIRegister_ovccoco(CL: TPSPascalCompiler);
12370: begin
12371: 'ovsetsize','LongInt').SetInt( 16);
12372: 'etSyntax','LongInt').SetInt( 0);
12373: 'etSymantic','LongInt').SetInt( 1);
12374: 'chCR','Char').SetString( #13);
12375: 'chLF','Char').SetString( #10);
12376: 'chLineSeparator','').SetString( chCR);
12377: SIRegister_TCocoError(CL);
12378: SIRegister_TCommentItem(CL);

```

```

12379:   SIRegister_TCommentList(CL);
12380:   SIRegister_TSymbolPosition(CL);
12381: TGenListType', '( glNever, glAlways, glOnError )
12382: TovBitSet', 'set of Integer
12383:   //PStartTable', '^TStartTable // will not work
12384: 'TovCharSet', 'set of AnsiChar
12385: TAfterGenListEvent', 'Procedure ( Sender : TObject; var PrintErrorCount : boolean)
12386: TCommentEvent', 'Procedure ( Sender : TObject; CommentList : TCommentList)
12387: TCustomErrorEvent', 'Function(Sender:TObject;const ErrorCode: longint;const Data:string): string
12388: TovErrorEvent', 'Procedure ( Sender : TObject; Error : TCocoError)
12389: TovErrorProc', 'Procedure ( ErrorCode : integer; Symbol : TSymbolP'
12390:   +osition; const Data : string; ErrorType : integer)
12391: TFailureEvent', 'Procedure ( Sender : TObject; NumErrors : integer)
12392: TGetCH', 'Function ( pos : longint) : char
12393: TStatusUpdateProc', 'Procedure ( Sender : TObject; Status: string; LineNum:integer)
12394:   SIRegister_TCocoRScanner(CL);
12395:   SIRegister_TCocoRGrammar(CL);
12396: '_EF','Char').SetString( #0);
12397: '_TAB','Char').SetString( #09);
12398: '_CR','Char').SetString( #13);
12399: '_LF','Char').SetString( #10);
12400: '_EL','').SetString( _CR);
12401: '_EOF','Char').SetString( #26);
12402: 'LineEnds','TCharSet').SetInt(ord(_CR) or ord(_LF) or ord(_EF));
12403: 'minErrDist','LongInt').SetInt( 2);
12404: Function ovPadL( S : string; ch : char; L : integer) : string
12405: end;
12406:
12407: TFormatSettings = record
12408:   CurrencyFormat: Byte;
12409:   NegCurrFormat: Byte;
12410:   ThousandSeparator: Char;
12411:   DecimalSeparator: Char;
12412:   CurrencyDecimals: Byte;
12413:   DateSeparator: Char;
12414:   TimeSeparator: Char;
12415:   ListSeparator: Char;
12416:   CurrencyString: string;
12417:   ShortDateFormat: string;
12418:   LongDateFormat: string;
12419:   TimeAMString: string;
12420:   TimePMString: string;
12421:   ShortTimeFormat: string;
12422:   LongTimeFormat: string;
12423:   ShortMonthNames: array[1..12] of string;
12424:   LongMonthNames: array[1..12] of string;
12425:   ShortDayNames: array[1..7] of string;
12426:   LongDayNames: array[1..7] of string;
12427:   TwoDigitYearCenturyWindow: Word;
12428: end;
12429:
12430: procedure SIRegister_OvcFormatSettings(CL: TPSPascalCompiler);
12431: begin
12432:   Function ovFormatSettings : TFormatSettings
12433: end;
12434:
12435: procedure SIRegister_ovcstr(CL: TPSPascalCompiler);
12436: begin
12437:   TOvcCharSet', 'set of Char
12438:   ovBTable', 'array[0..255] of Byte
12439:   //BTable = array[0..{$IFDEF UNICODE}{$IFDEF
HUGE_UNICODE_BMTABLE}$FFFF{$ELSE}$FF{$ENDIF}{$ELSE}$FF{$ENDIF}] of Byte;
12440:   Function BinaryBPChar( Dest : PChar; B : Byte) : PChar
12441:   Function BinaryLPChar( Dest : PChar; L : LongInt) : PChar
12442:   Function BinaryWPChar( Dest : PChar; W : Word) : PChar
12443:   Procedure BMMakeTable( MatchString : PChar; var BT : ovBTable)
12444:   Function BMSearch(var Buffer,BufLength:Cardinal;var BT:ovBTable; MatchString:PChar;var Pos:Cardinal):Bool;
12445:   Function BMSearchUC(var Buffer,BufLength:Card;var BT:ovBTable;MatchString:PChar;var Pos: Card):Boolean
12446:   Function CharStrPChar( Dest : PChar; C : Char; Len : Cardinal) : PChar
12447:   Function DetabPChar( Dest : PChar; Src : PChar; TabSize : Byte) : PChar
12448:   Function HexBPChar( Dest : PChar; B : Byte) : PChar
12449:   Function HexLPChar( Dest : PChar; L : LongInt) : PChar
12450:   Function HexPtrPChar( Dest : PChar; P : TObject) : PChar
12451:   Function HexWPChar( Dest : PChar; W : Word) : PChar
12452:   Function LoCaseChar( C : Char) : Char
12453:   Function OctalLPChar( Dest : PChar; L : LongInt) : PChar
12454:   Function StrChDeletePrim( P : PChar; Pos : Cardinal) : PChar
12455:   Function StrChInsertPrim( Dest : PChar; C : Char; Pos: Cardinal) : PChar
12456:   Function StrChPos( P : PChar; C : Char; var Pos : Cardinal) : Boolean
12457:   Procedure StrInsertChars( Dest : PChar; Ch : Char; Pos, Count : Word)
12458:   Function StrStCopy( Dest, S : PChar; Pos, Count : Cardinal) : PChar
12459:   Function StrStDeletePrim( P : PChar; Pos, Count : Cardinal) : PChar
12460:   Function StrStInsert( Dest, S1, S2 : PChar; Pos : Cardinal) : PChar
12461:   Function StrStInsertPrim( Dest, S : PChar; Pos : Cardinal) : PChar
12462:   Function StrStPos( P, S : PChar; var Pos : Cardinal) : Boolean
12463:   Function StrToLongPChar( S : PChar; var I : LongInt) : Boolean
12464:   Procedure TrimAllSpacesPChar( P : PChar)
12465:   Function TrimEmbeddedZeros( const S : string) : string
12466:   Procedure TrimEmbeddedZerosPChar( P : PChar)

```



```

12467: Function TrimTrailPrimPChar( S : PChar) : PChar
12468: Function TrimTrailPChar( Dest, S : PChar) : PChar
12469: Function TrimTrailingZeros( const S : string) : string
12470: Procedure TrimTrailingZerosPChar( P : PChar)
12471: Function UpCaseChar( C : Char) : Char
12472: Function ovcCharInSet( C : Char; const CharSet : TOvcCharSet) : Boolean
12473: Function ovc32StringIsCurrentCodePage( const S : WideString) : Boolean;
12474: //Function ovc32StringIsCurrentCodePage1( const S:PWideChar; CP : Cardinal) : Boolean;
12475: end;
12476:
12477: procedure SIRegister_AfUtils(CL: TPSPascalCompiler);
12478: begin
12479:   //PraiseFrame, '^TraiseFrame // will not work
12480:   TraiseFrame', record NextRaise : PRaiseFrame; ExceptAddr : ___Poin'
12481:   +'ter; ExceptObject : TObject; ExceptionRecord : PExceptionRecord; end
12482:   Procedure SafeCloseHandle( var Handle : THandle)
12483:   Procedure ExchangeInteger( X1, X2 : Integer)
12484:   Procedure FillInteger( const Buffer, Size, Value : Integer)
12485:   Function LongMulDiv( Mult1, Mult2, Div1 : Longint) : Longint
12486:   Function afCompareMem( P1, P2 : TObject; Length : Integer) : Boolean
12487:
12488:   FILENAME_ADVAPI32 = 'ADVAPI32.DLL';
12489:   function AbortSystemShutdown; external advapi32 name 'AbortSystemShutdownW';
12490:   function AbortSystemShutdown(lpMachineName: PKOLChar): BOOL; stdcall;
12491:   function AccessCheckAndAuditAlarm(SubsystemName: PKOLChar;
12492:   HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12493:   SecurityDescriptor: PSecurityDescriptor; DesiredAccess: DWORD;
12494:   const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12495:   var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12496:   function AccessCheckByTypeAndAuditAlarm(SubsystemName: PKOLChar;
12497:   HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12498:   SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12499:   AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12500:   ObjectTypeInfoLength: DWORD; const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12501:   var GrantedAccess: DWORD; var AccessStatus, pfGenerateOnClose: BOOL): BOOL; stdcall;
12502:   function AccessCheckByTypeResultListAndAuditAlarm(SubsystemName: PKOLChar;
12503:   HandleId: Pointer; ObjectTypeName, ObjectName: PKOLChar;
12504:   SecurityDescriptor: PSecurityDescriptor; PrincipalSelfSid: PSID; DesiredAccess: DWORD;
12505:   AuditType: AUDIT_EVENT_TYPE; Flags: DWORD; ObjectTypeList: PObjectTypeList;
12506:   ObjectTypeInfoLength: DWORD; const GenericMapping: TGenericMapping; ObjectCreation: BOOL;
12507:   var GrantedAccess: DWORD; var AccessStatusList: DWORD; var pfGenerateOnClose: BOOL): BOOL; stdcall;
12508:   function BackupEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12509:   function ClearEventLog(hEventLog: THandle; lpBackupFileName: PKOLChar): BOOL; stdcall;
12510:   function CreateProcessAsUser(hToken: THandle; lpApplicationName: PKOLChar;
12511:   lpCommandLine: PKOLChar; lpProcessAttributes: PSecurityAttributes;
12512:   lpThreadAttributes: PSecurityAttributes; bInheritHandles: BOOL;
12513:   dwCreationFlags: DWORD; lpEnvironment: Pointer; lpCurrentDirectory: PKOLChar;
12514:   const lpStartupInfo: TStartupInfo; var lpProcessInformation: TProcessInformation): BOOL; stdcall;
12515:   function GetCurrentHwProfile(var lpHwProfileInfo: THWPProfileInfo): BOOL; stdcall;
12516:   function GetFileSecurity(lpFileName: PKOLChar; RequestedInformation: SECURITY_INFORMATION;
12517:   pSecurityDescriptor: PSecurityDescriptor; nLength: DWORD; var lpnLengthNeeded: DWORD): BOOL; stdcall;
12518:   function GetUserName(lpBuffer: PKOLChar; var nSize: DWORD): BOOL; stdcall;
12519:   function InitiateSystemShutdown(lpMachineName, lpMessage: PKOLChar;
12520:   dwTimeout: DWORD; bForceAppsClosed, bRebootAfterShutdown: BOOL): BOOL; stdcall;
12521:   function LogonUser(lpszUsername, lpszDomain, lpszPassword: PKOLChar;
12522:   dwLogonType, dwLogonProvider: DWORD; var phToken: THandle): BOOL; stdcall;
12523:   function LookupAccountName(lpSystemName, lpAccountName: PKOLChar;
12524:   Sid: PSID; var cbSid: DWORD; ReferencedDomainName: PKOLChar;
12525:   var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12526:   function LookupAccountSid(lpSystemName: PKOLChar; Sid: PSID;
12527:   Name: PKOLChar; var cbName: DWORD; ReferencedDomainName: PKOLChar;
12528:   var cbReferencedDomainName: DWORD; var peUse: SID_NAME_USE): BOOL; stdcall;
12529:   function LookupPrivilegeDisplayName(lpSystemName, lpName: PKOLChar;
12530:   lpDisplayName: PKOLChar; var cbDisplayName, lpLanguageId: DWORD): BOOL; stdcall;
12531:   function LookupPrivilegeName(lpSystemName: PKOLChar;
12532:   var lpLuid: TLargeInteger; lpName: PKOLChar; var cbName: DWORD): BOOL; stdcall;
12533:   function LookupPrivilegeValue(lpSystemName, lpName: PKOLChar;
12534:   var lpLuid: TLargeInteger): BOOL; stdcall;
12535:   function ObjectCloseAuditAlarm(SubsystemName: PKOLChar;
12536:   HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12537:   function ObjectDeleteAuditAlarm(SubsystemName: PKOLChar;
12538:   HandleId: Pointer; GenerateOnClose: BOOL): BOOL; stdcall;
12539:   function ObjectOpenAuditAlarm(SubsystemName: PKOLChar; HandleId: Pointer;
12540:   ObjectTypeName: PKOLChar; ObjectName: PKOLChar; pSecurityDescriptor: PSecurityDescriptor;
12541:   ClientToken: THandle; DesiredAccess, GrantedAccess: DWORD;
12542:   var Privileges: TPrivilegeSet; ObjectCreation, AccessGranted: BOOL;
12543:   var GenerateOnClose: BOOL): BOOL; stdcall;
12544:   function ObjectPrivilegeAuditAlarm(SubsystemName: PKOLChar;
12545:   HandleId: Pointer; ClientToken: THandle; DesiredAccess: DWORD;
12546:   var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12547:   function OpenBackupEventLog(lpUNCServerName, lpFileName: PKOLChar): THandle; stdcall;
12548:   function OpenEventLog(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12549:   function PrivilegedServiceAuditAlarm(SubsystemName, ServiceName: PKOLChar;
12550:   ClientToken: THandle; var Privileges: TPrivilegeSet; AccessGranted: BOOL): BOOL; stdcall;
12551:   function ReadEventLog(hEventLog: THandle; dwReadFlags, dwRecordOffset: DWORD;
12552:   lpBuffer: Pointer; nNumberOfBytesToRead: DWORD;
12553:   var pnBytesRead, pnMinNumberOfBytesNeeded: DWORD): BOOL; stdcall;
12554:   function RegConnectRegistry(lpMachineName: PKOLChar; hKey: HKEY;
12555:   var phkResult: HKEY): Longint; stdcall;

```

```

12556:     function RegCreateKey(hKey: HKEY; lpSubKey: PKOLChar;
12557:         var phkResult: HKEY): Longint; stdcall;
12558:     function RegCreateKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12559:         Reserved: DWORD; lpClass: PKOLChar; dwOptions: DWORD; samDesired: REGSAM;
12560:         lpSecurityAttributes: PSecurityAttributes; var phkResult: HKEY;
12561:         lpdwDisposition: PDWORD): Longint; stdcall;
12562:     function RegDeleteKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12563:     function RegDeleteValue(hKey: HKEY; lpValueName: PKOLChar): Longint; stdcall;
12564:     function RegEnumKeyEx(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar;
12565:         var lpcbName: DWORD; lpReserved: Pointer; lpClass: PKOLChar;
12566:         lpcbClass: PDWORD; lpftLastWriteTime: PFileTime): Longint; stdcall;
12567:     function RegEnumKey(hKey: HKEY; dwIndex: DWORD; lpName: PKOLChar; cbName: DWORD): Longint; stdcall;
12568:     function RegEnumValue(hKey: HKEY; dwIndex: DWORD; lpValueName: PKOLChar;
12569:         var lpcbValueName: DWORD; lpReserved: Pointer; lpType: PDWORD;
12570:         lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12571:     function RegLoadKey(hKey: HKEY; lpSubKey, lpFile: PKOLChar): Longint; stdcall;
12572:     function RegOpenKey(hKey: HKEY; lpSubKey: PKOLChar; var phkResult: HKEY): Longint; stdcall;
12573:     function RegOpenKeyEx(hKey: HKEY; lpSubKey: PKOLChar;
12574:         ulOptions: DWORD; samDesired: REGSAM; var phkResult: HKEY): Longint; stdcall;
12575:     function RegQueryInfoKey(hKey: HKEY; lpClass: PKOLChar;
12576:         lpcbClass: PDWORD; lpReserved: Pointer;
12577:         lpcbSubKeys, lpcbMaxSubKeyLen, lpcbMaxClassLen, lpcbValues,
12578:         lpcbMaxValueNameLen, lpcbMaxValueLen, lpcbSecurityDescriptor: PDWORD;
12579:         lpftLastWriteTime: PFileTime): Longint; stdcall;
12580:     function RegQueryMultipleValues(hKey: HKEY; var Vallist;
12581:         NumVals: DWORD; lpValueBuf: PKOLChar; var ldwTotSize: DWORD): Longint; stdcall;
12582:     function RegQueryValue(hKey: HKEY; lpSubKey: PKOLChar;
12583:         lpValue: PKOLChar; var lpcbValue: Longint): Longint; stdcall;
12584:     function RegQueryValueEx(hKey: HKEY; lpValueName: PKOLChar;
12585:         lpReserved: Pointer; lpType: PDWORD; lpData: PByte; lpcbData: PDWORD): Longint; stdcall;
12586:     function RegReplaceKey(hKey: HKEY; lpSubKey: PKOLChar;
12587:         lpNewFile: PKOLChar; lpOldFile: PKOLChar): Longint; stdcall;
12588:     function RegRestoreKey(hKey: HKEY; lpFile: PKOLChar; dwFlags: DWORD): Longint; stdcall;
12589:     function RegSaveKey(hKey: HKEY; lpFile: PKOLChar;
12590:         lpSecurityAttributes: PSecurityAttributes): Longint; stdcall;
12591:     function RegSetValue(hKey: HKEY; lpSubKey: PKOLChar;
12592:         dwType: DWORD; lpData: PKOLChar; cbData: DWORD): Longint; stdcall;
12593:     function RegSetValueEx(hKey: HKEY; lpValueName: PKOLChar;
12594:         Reserved: DWORD; dwType: DWORD; lpData: Pointer; cbData: DWORD): Longint; stdcall;
12595:     function RegUnLoadKey(hKey: HKEY; lpSubKey: PKOLChar): Longint; stdcall;
12596:     function RegisterEventSource(lpUNCServerName, lpSourceName: PKOLChar): THandle; stdcall;
12597:     function ReportEvent(hEventLog: THandle; wType, wCategory: Word;
12598:         dwEventID: DWORD; lpUserSid: Pointer; wNumStrings: Word;
12599:         dwDataSize: DWORD; lpStrings, lpRawData: Pointer): BOOL; stdcall;
12600:     function SetFileSecurity(lpFileName: PKOLChar; SecurityInformation: SECURITY_INFORMATION;
12601:         pSecurityDescriptor: PSecurityDescriptor): BOOL; stdcall;
12602:
12603:     Function wAddAtom( lpString : PKOLChar ) : ATOM
12604:     Function wBeginUpdateResource( pFileName : PKOLChar; bDeleteExistingResources : BOOL ) : THandle
12605:     //Function wCallNamedPipe( lpNamedPipeName : PKOLChar; lpInBuffer : Pointer; nInBufferSize : DWORD;
12606:         lpOutBuffer : Pointer; nOutBufferSize : DWORD; var lpBytesRead : DWORD; nTimeOut : DWORD ) : BOOL
12607:     //Function wCommConfigDialog( lpszName : PKOLChar; hWnd : HWND; var lpCC : TCommConfig ) : BOOL
12608:     Function wCompareString( Locale : LCID; dwCmpFlags : DWORD; lpString1 : PKOLChar; cchCount1 : Integer;
12609:         lpString2 : PKOLChar; cchCount2 : Integer ) : Integer
12610:     Function wCopyFile( lpExistingFileName, lpNewFileName : PKOLChar; bFailIfExists : BOOL ) : BOOL
12611:     //Function wCopyFileEx( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
12612:         TFNProgressRoutine; lpData : Pointer; pbCancel : PBool; dwCopyFlags : DWORD ) : BOOL
12613:     Function wCreateDirectory( lpPathName : PKOLChar; lpSecurityAttributes : PSecurityAttributes ) : BOOL
12614:     Function wCreateDirectoryEx( lpTemplateDirectory,
12615:         lpNewDirectory: PKOLChar; lpSecAttrib: PSecurityAttribs ): BOOL;
12616:     Function wCreateEvent( lpEventAttribs: PSecurityAttrib; bManualReset,
12617:         bInitialState: BOOL; lpName: PKOLChar ): THandle;
12618:     Function wCreateFile( lpFileName : PKOLChar; dwDesiredAccess, dwShareMode : DWORD; lpSecurityAttributes :
12619:         PSecurityAttributes; dwCreationDisposition, dwFlagsAndAttributes: DWORD; hTemplateFile: THandle ): THandle
12620:     Function wCreateFileMapping( hFile : THandle; lpFileMappingAttributes : PSecurityAttributes; flProtect,
12621:         dwMaximumSizeHigh, dwMaximumSizeLow : DWORD; lpName : PKOLChar ) : THandle
12622:     Function wCreateHardLink( lpFileName,
12623:         lpExistingFileName: PKOLChar; lpSecurityAttributes: PSecurityAttributes ): BOOL
12624:     Function
12625:         CreateMailslot( lpName: PKOLChar; MaxMessSize: DWORD; lReadTimeout: DWORD; lpSecurityAttrib: PSecurityAttributes ): THandle;
12626:     Function wCreateNamedPipe( lpName : PKOLChar; dwOpenMode, dwPipeMode, nMaxInstances, nOutBufferSize,
12627:         nInBufferSize, nDefaultTimeOut : DWORD; lpSecurityAttributes : PSecurityAttributes ) : THandle
12628:     //Function CreateProcess( lpApplicationName : PKOLChar; lpCommandLine : PKOLChar; lpProcessAttributes,
12629:         lpThreadAttributes : PSecurityAttributes; bInheritHandles : BOOL; dwCreationFlags : DWORD; lpEnvironment :
12630:         Pointer; lpCurrentDirectory: PKOLChar; const lpStartupInfo: TStartupInfo; var
12631:         lpProcessInfo: TProcessInformation ): BOOL
12632:     Function wCreateSemaphore( lpSemaphoreAttributes: PSecurityAttributes; lInitialCount, lMaximumCount :
12633:         Longint; lpName : PKOLChar ) : THandle
12634:     Function
12635:         wCreateWaitableTimer( lpTimerAttributes: PSecurityAttribs; bManualReset: BOOL; lpTimerName: PKOLChar ): THandle;
12636:     Function wDefineDosDevice( dwFlags : DWORD; lpDeviceName, lpTargetPath : PKOLChar ) : BOOL
12637:     Function wDeleteFile( lpFileName : PKOLChar ) : BOOL
12638:     Function wEndUpdateResource( hUpdate : THandle; fDiscard : BOOL ) : BOOL
12639:     //Function
12640:         wEnumCalendarInfo( lpCalInfEnumProc: TFNCalInfEnumProc; Locale: LCID; Calendar: CALID; CalType: CALTYPE ): BOOL;
12641:     //Function wEnumDateFormats( lpDateFmtEnumProc: TFNDateFmtEnumProc; Locale : LCID; dwFlags : DWORD ) : BOOL
12642:     // Function wEnumResourceLanguages( hModule: HMODULE; lpType,
12643:         lpName: PChar; lpEnumFunc: ENUMRESLANGPROC; lParam: Longint : BOOL )
12644:     //Function
12645:         wEnumResourceNames( hModule: HMODULE; lpType: PKOLChar; lpEnumFunc: ENUMRESNAMEPROC; lParam: Longint ) : BOOL;

```

```

12628: //Function wEnumResourceTypes( hModule:HMODULE; lpEnumFunc:ENUMRESTYPEPROC;lpParam:Longint):BOOL;
12629: //Function wEnumSystemCodePages( lpCodePageEnumProc : TFNCodepageEnumProc; dwFlags : DWORD) : BOOL
12630: //Function wEnumSystemLocales( lpLocaleEnumProc : TFNLocaleEnumProc; dwFlags : DWORD) : BOOL
12631: //Function wEnumTimeFormats( lpTimeFmtEnumProc:TFNTimeFmtEnumProc;Locale:LCID;dwFlags:DWORD):BOOL;
12632: Function wExpandEnvironmentStrings( lpSrc : PKOLChar; lpDst : PKOLChar; nSize : DWORD) : DWORD
12633: Procedure wFatalAppExit( uAction : UINT; lpMessageText : PKOLChar)
12634: //Function wFillConsoleOutputCharacter( hConsoleOutput : THandle; cCharacter : KOLChar; nLength : DWORD;
dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12635: Function wFindAtom( lpString : PKOLChar) : ATOM
12636: Function
wFindFirstChangeNotification(lpPathName:PKOLChar;bWatchSubtree:BOOL;dwNotifyFilter:DWORD):THandle;
12637: Function wFindFirstFile( lpFileName : PKOLChar; var lpFindFileData : TWIN32FindData) : THandle
12638: //Function wFindFirstFileEx( lpFileName : PKOLChar; fInfoLevelId : TFindInfoLevels; lpFindFileData :
Pointer; fSearchOp : TFindSearchOps; lpSearchFilter : Pointer; dwAdditionalFlags : DWORD) : BOOL
12639: Function wFindNextFile( hFindFile : THandle; var lpFindFileData : TWIN32FindData) : BOOL
12640: Function wFindResource( hModule : HMODULE; lpName, lpType : PKOLChar) : HRSRC
12641: Function wFindResourceEx( hModule : HMODULE; lpType, lpName : PKOLChar; wLanguage : Word) : HRSRC
12642: Function
wFoldString(dwMapFlags:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Integer):Integer;
12643: //Function wFormatMessage( dwFlags : DWORD; lpSource : Pointer; dwMessageId : DWORD; dwLanguageId :
DWORD; lpBuffer : PKOLChar; nSize : DWORD; Arguments : Pointer) : DWORD
12644: Function wFreeEnvironmentStrings( EnvBlock : PKOLChar) : BOOL
12645: Function wGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12646: Function wGetBinaryType( lpApplicationName : PKOLChar; var lpBinaryType : DWORD) : BOOL
12647: Function wGetCommandLine : PKOLChar
12648: //Function wGetCompressedFileSize( lpFileName : PKOLChar; lpFileSizeHigh : PDWORD) : DWORD
12649: Function wGetComputerName( lpBuffer : PKOLChar; var nSize : DWORD) : BOOL
12650: Function wGetConsoleTitle( lpConsoleTitle : PKOLChar; nSize : DWORD) : DWORD
12651: //Function wGetCurrencyFormat( Locale : LCID; dwFlags : DWORD; lpValue : PKOLChar; lpFormat :
PCurrencyFmt; lpCurrencyStr : PKOLChar; cchCurrency : Integer) : Integer
12652: Function wGetCurrentDirectory( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12653: //Function wGetDateFormat( Locale : LCID; dwFlags : DWORD; lpDate : PSystemTime; lpFormat : PKOLChar;
lpDateStr : PKOLChar; cchDate : Integer) : Integer
12654: //Function wGetDefaultCommConfig( lpSzName:PKOLChar;var lpCC : TCommConfig;var lpdwSize:DWORD):BOOL
12655: Function wGetDiskFreeSpace( lpRootPathName : PKOLChar; var lpSectorsPerCluster, lpBytesPerSector,
lpNumberOfFreeClusters, lpTotalNumberOfClusters : DWORD) : BOOL
12656: //Function wGetDiskFreeSpaceEx( lpDirectoryName : PKOLChar; var lpFreeBytesAvailableToCaller,
lpTotalNumberOfBytes, lpTotalNumberOfFreeBytes : PLargeInteger) : BOOL
12657: Function wGetDriveType( lpRootPathName : PKOLChar) : UINT
12658: Function wGetEnvironmentStrings : PKOLChar
12659: Function wGetEnvironmentVariable( lpName : PKOLChar; lpBuffer : PKOLChar; nSize : DWORD) : DWORD;
12660: Function wGetFileAttributes( lpFileName : PKOLChar) : DWORD
12661: //Function
wGetFileAttributesEx(lpFileName:PKOLChar;fInfoLevelId:TGetFileExInfoLevs;lpFileInform:Pointer):BOOL;
12662: Function wGetFullPathName(lpFileName:PKOLChar;nBufferLength:WORD;lpBuffer:PKOLChar;var
lpFilePart:PKOLChar):DWORD;
12663: //Function wGetLocaleInfo(Locale:LCID; LCTYPE:LCTYPE;lpLCData:PKOLChar;cchData:Integer) : Integer
12664: Function wGetLogicalDriveStrings( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12665: Function wGetModuleFileName( hModule : HINST; lpFileName : PKOLChar; nSize : DWORD) : DWORD
12666: Function wGetModuleHandle( lpModuleName : PKOLChar) : HMODULE
12667: //Function wGetNamedPipeHandleState( hNamedPipe : THandle; lpState, lpCurInstances, lpMaxCollectionCount,
lpCollectDataTimeout : PDWORD; lpUserName : PKOLChar; nMaxUserNameSize : DWORD) : BOOL
12668: //Function wGetNumberFormat( Locale : LCID; dwFlags:DWORD; lpValue:PKOLChar; lpFormat:PNumberFmt;
lpNumberStr : PKOLChar; cchNumber : Integer) : Integer
12669: Function wGetPrivateProfileInt(lpAppName,lpKeyName:PKOLChar;nDefault:Integer;lpFileName:PKOLChar):UINT;
12670: Function
wGetPrivateProfileSection(lpAppName:PKOLChar;lpRetrStr:PKOLChar;nSize:DWORD;pFileName:PKOLChar):DWORD;
12671: Function wGetPrivateProfileSectionNames(lpSzReturnBuffer:PKOLChar;nSize:DWORD;lpFileName:PKOLChar):DWORD;
12672: Function wGetPrivateProfileString( lpAppName, lpKeyName, lpDefault : PKOLChar;lpReturnedStr: PKOLChar;
nSize:DWORD; lpFileName : PKOLChar) : DWORD
12673: Function wGetProfileInt( lpAppName, lpKeyName : PKOLChar; nDefault : Integer) : UINT
12674: Function wGetProfileSection( lpAppName : PKOLChar; lpReturnedString : PKOLChar; nSize : DWORD) : DWORD
12675: Function wGetProfileString(lpAppName,lpKeyName,
lpDefault:PKOLChar;lpReturnedStr:PKOLChar;nSize:DWORD):DWORD;
12676: Function wGetShortPathName( lpSzLongPath:PKOLChar;lpSzShortPath: PKOLChar; cchBuffer : DWORD) : DWORD
12677: //Procedure wGetStartupInfo( var lpStartupInfo : TStartupInfo)
12678: //Function wGetStringTypeEx(Locale:LCID; dwInfoType:DWORD;lpSrcStr:PKOLChar;cchSrc:Integer;var
lpCharType):BOOL
12679: Function wGetSystemDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12680: Function wGetTempFileName( lpPathName, lpPrefixString: PKOLChar;uUnique:UINT;lpTempFileName:PKOLChar):UINT
12681: Function wGetTempPath( nBufferLength : DWORD; lpBuffer : PKOLChar) : DWORD
12682: //Function
wGetTimeFormat(Loc:LCID;dwFlgs:DWORD;lpTime:PSystemTime;lpFrm:PKOLChar;lpTimeStr:PKOLChar;cTime:Int):Int
12683: //Function wGetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
12684: //Function GetVolumeInformation( lpRootPathName : PKOLChar; lpVolumeNameBuffer : PKOLChar; nVolumeNameSize
: DWORD; lpVolumeSerialNumber : PDWORD; var lpMaximumComponentLength, lpFileSystemFlags : DWORD;
lpFileSystemNameBuffer : PKOLChar; nFileSystemNameSize : DWORD) : BOOL
12685: Function wGetWindowsDirectory( lpBuffer : PKOLChar; uSize : UINT) : UINT
12686: Function wGlobalAddAtom( lpString : PKOLChar) : ATOM
12687: Function wGlobalFindAtom( lpString : PKOLChar) : ATOM
12688: Function wGlobalGetAtomName( nAtom : ATOM; lpBuffer : PKOLChar; nSize : Integer) : UINT
12689: Function wIsBadStringPtr( lpSz : PKOLChar; ucchMax : UNT) : BOOL
12690: Function
wLCMapString(Loc:LCID;dwMapFlgs:DWORD;lpSrcStr:PKOLChar;cchSrc:Int;lpDestStr:PKOLChar;cchDest:Int):Int;
12691: Function wLoadLibrary( lpLibFileName : PKOLChar) : HMODULE
12692: Function wLoadLibraryEx( lpLibFileName : PKOLChar; hFile : THandle; dwFlags : DWORD) : HMODULE
12693: Function wMoveFile( lpExistingFileName, lpNewFileName : PKOLChar) : BOOL
12694: Function wMoveFileEx( lpExistingFileName, lpNewFileName : PKOLChar; dwFlags : DWORD) : BOOL
12695: //Function wMoveFileWithProgress( lpExistingFileName, lpNewFileName : PKOLChar; lpProgressRoutine :
TFNProgressRoutine; lpData : Pointer; dwFlags : DWORD) : BOOL

```



```

12696: Function wOpenEvent( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName:PKOLChar ) : THandle
12697: Function wOpenFileMapping( dwDesiredAccess : DWORD; bInheritHandle:BOOL;lpName: PKOLChar):THandle
12698: Function wOpenMutex( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar) : THandle
12699: Function wOpenSemaphore( dwDesiredAccess : DWORD; bInheritHandle : BOOL; lpName : PKOLChar):THandle
12700: Function wOpenWaitableTimer(dwDesiredAccess:DWORD;bInheritHandle:BOOL;lpTimerName:PKOLChar):THandle
12701: Procedure wOutputDebugString( lpOutputString : PKOLChar)
12702: //Function wPeekConsoleInput(hConsoleInput:THandle;var lpBuffer:TInputRecord;nLength:DWORD;var
lpNumberOfEventsRead:DWORD):BOOL;
12703: Function wQueryDosDevice( lpDeviceName : PKOLChar; lpTargetPath : PKOLChar; ucchMax : DWORD) : DWORD
12704: //Function wQueryRecoveryAgents(p1:PKOLChar;var p2:Pointer;var p3:TRecoveryAgentInformation):DWORD
12705: //Function wReadConsole( hConsoleInput : THandle; lpBuffer : Pointer; nNumberOfCharsToRead : DWORD; var
lpNumberOfCharsRead : DWORD; lpReserved : Pointer) : BOOL
12706: //Function wReadConsoleInput(hConsInp:THandle;var lpBuf:TInpRec;nLength:DWORD;var
lpNumbOfEventsRead:DWORD):BOOL;
12707: //Function wReadConsoleOutput( hConsoleOutput : THandle; lpBuffer : Pointer; dwBufferSize, dwBufferCoord
: TCoord; var lpReadRegion : TSmallRect) : BOOL
12708: //Function wReadConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
DWORD; dwReadCoord : TCoord; var lpNumberOfCharsRead : DWORD) : BOOL
12709: Function wRemoveDirectory( lpPathName : PKOLChar) : BOOL
12710: //Function wScrollConsoleScreenBuffer( hConsoleOutput : THandle; const lpScrollRectangle : TSmallRect;
lpClipRectangle : PSmallRect; dwDestinationOrigin : TCoord; var lpFill : TCharInfo) : BOOL
12711: Function wSearchPath( lpPath,lpFileName,lpExtension:PKOLChar;nBufferLength:DWORD;lpBuffer:PKOLChar;var
lpFilePart:PKOLChar):DWORD;
12712: Function wSetComputerName( lpComputerName : PKOLChar) : BOOL
12713: Function wSetConsoleTitle( lpConsoleTitle : PKOLChar) : BOOL
12714: Function wSetCurrentDirectory( lpPathName : PKOLChar) : BOOL
12715: //Function wSetDefaultCommConfig( lpszName : PKOLChar; lpCC : PCommConfig; dwSize : DWORD) : BOOL
12716: Function wSetEnvironmentVariable( lpName, lpValue : PKOLChar) : BOOL
12717: Function wSetFileAttributes( lpFileName : PKOLChar; dwFileAttributes : DWORD) : BOOL
12718: //Function wSetLocaleInfo( Locale : LCID; LCTYPE : LCTYPE; lpLCData : PKOLChar) : BOOL
12719: Function wSetVolumeLabel( lpRootPathName : PKOLChar; lpVolumeName : PKOLChar) : BOOL
12720: //Function wUpdateResource(hUpdate:THandle;lpType,
lpName:PKOLChar;wLanguage:Word;lpData:Ptr;cbData:DWORD):BOOL
12721: Function wVerLanguageName( wLang : DWORD; szLang : PKOLChar; nSize : DWORD) : DWORD
12722: Function wWaitNamedPipe( lpNamedPipeName : PKOLChar; nTimeOut : DWORD) : BOOL
12723: //Function wWriteConsole( hConsoleOutput : THandle; const lpBuffer : Pointer; nNumberOfCharsToWrite :
DWORD; var lpNumberOfCharsWritten : DWORD; lpReserved : Pointer) : BOOL
12724: //Function wWriteConsoleInput( hConsoleInput : THandle; const lpBuffer : TInputRecord; nLength : DWORD;
var lpNumberOfEventsWritten : DWORD) : BOOL
12725: //Function wWriteConsoleOutput(hConsoleOutput:THandle; lpBuffer:Pointer; dwBufferSize,dwBufferCoord :
TCoord; var lpWriteRegion : TSmallRect) : BOOL
12726: //Function wWriteConsoleOutputCharacter( hConsoleOutput : THandle; lpCharacter : PKOLChar; nLength :
DWORD; dwWriteCoord : TCoord; var lpNumberOfCharsWritten : DWORD) : BOOL
12727: Function wWritePrivateProfileSection( lpAppName, lpString, lpFileName : PKOLChar) : BOOL
12728: Function wWritePrivateProfileString( lpAppName, lpKeyName, lpString, lpFileName : PKOLChar) : BOOL
12729: Function wWriteProfileSection( lpAppName, lpString : PKOLChar) : BOOL
12730: Function wWriteProfileString( lpAppName, lpKeyName, lpString : PKOLChar) : BOOL
12731: Function wlstrcat( lpString1, lpString2 : PKOLChar) : PKOLChar
12732: Function wlstrcmp( lpString1, lpString2 : PKOLChar) : Integer
12733: Function wlstrcmpi( lpString1, lpString2 : PKOLChar) : Integer
12734: Function wlstrcpy( lpString1, lpString2 : PKOLChar) : PKOLChar
12735: Function wlstrncpy( lpString1, lpString2 : PKOLChar; iMaxLength : Integer) : PKOLChar
12736: Function wlstrlen( lpString : PKOLChar) : Integer
12737: //Function wMultinetGetConnectionPerformance( lpNetResource : PNetResource; lpNetConnectInfoStruc :
PNetConnectInfoStruct) : DWORD
12738: //Function wNetAddConnection2(var lpNetResource:TNetResource;lpPassword,
lpUserName:PKOLChar;dwFlags:DWORD):DWORD;
12739: //Function wNetAddConnection3( hwndOwner : HWND; var lpNetResource:TNetResource;lpPassword,
lpUserName:PKOLChar; dwFlags : DWORD) : DWORD
12740: Function wNetAddConnection( lpRemoteName, lpPassword, lpLocalName : PKOLChar) : DWORD
12741: Function wNetCancelConnection2( lpName : PKOLChar; dwFlags : DWORD; fForce : BOOL) : DWORD
12742: Function wNetCancelConnection( lpName : PKOLChar; fForce : BOOL) : DWORD
12743: //Function wNetConnectionDialog1( var lpConnDlgStruct : TConnectDlgStruct) : DWORD
12744: //Function wNetDisconnectDialog1( var lpConnDlgStruct : TDiscDlgStruct) : DWORD
12745: //Function wNetEnumResource(hEnum:THandle;var lpCount:DWORD;lpBuffer:Ptr;var lpBufferSize:DWORD):DWORD;
12746: Function wNetGetConnection(lpLocalName:PKOLChar;lpRemoteName:PKOLChar; var lpnLength:DWORD):DWORD;
12747: Function wNetGetLastError( var lpError : DWORD; lpErrorBuf : PKOLChar; nErrorBufSize : DWORD; lpNameBuf
: PKOLChar; nNameBufSize : DWORD) : DWORD
12748: //Function wNetGetNetworkInformation(lpProvider:PKOLChar;var lpNetInfoStruct:TNetInfoStruct):DWORD;
12749: Function wNetGetProviderName(dwNetType:DWORD;lpProviderName:PKOLChar;var lpBufferSize:DWORD):DWORD;
12750: //Function wNetGetResourceParent(lpNetResource:PNetResource;lpBuffer:Pointer;var cbBuffer:DWORD):DWORD;
12751: //Function wNetGetUniversalName(lpLocalPath:PKOLChar;dwInfoLevel:DWORD;lpBuffer:Ptr;var
lpBufferSize:DWORD):DWORD;
12752: Function wNetGetUser( lpName : PKOLChar; lpUserName : PKOLChar; var lpnLength : DWORD) : DWORD
12753: //Function wNetOpenEnum(dwScope,dwType,dwUsage:DWORD;lpNetResource:PNetRes;var lpEnum:THandle):DWORD;
12754: //Function wNetSetConnection( lpName : PKOLChar; dwProperties : DWORD; pvValues : Pointer) : DWORD
12755: //Function wNetUseConnection(hwndOwner:HWND;var
lpNetResource:TNetResource;lpUserID:PKOLChar;lpPassword:PKOLChar; dwFlags:DWORD;lpAccessName:PKOLChar;var
lpBufferSize:DWORD;var lpResult:DWORD) : DWORD
12756: Function wGetFileVersionInfo(lptstrFilename:PKOLChar;dwHandle,dwLen:DWORD;lpData:Pointer):BOOL;
12757: Function wGetFileVersionInfoSize( lptstrFilename : PKOLChar; var lpdwHandle : DWORD) : DWORD
12758: Function wVerFindFile( uFlags : DWORD; szFileName, szWinDir, szAppDir, szCurDir : PKOLChar; var
lpuCurDirLen : UINT; szDestDir : PKOLChar; var lpuDestDirLen : UINT) : DWORD
12759: Function wVerInstallFile( uFlags : DWORD; szSrcFileName, szDestFileName, szSrcDir, szDestDir, szCurDir,
szTmpFile : PKOLChar; var lpuTmpFileLen : UINT) : DWORD
12760: //Function wVerQueryValue(pBlock:Pointer;lpSubBlock:PKOLChar;var lplpBuffer:Ptr;var puLen:UINT):BOOL;
12761: //Function wGetPrivateProfileStruct(lpszSection,
lpSzKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;
12762: //Function wWritePrivateProfileStruct(lpszSection,
lpSzKey:PKOLChar;lpStruct:Ptr;uSizeStruct:UINT;szFile:PKOLChar):BOOL;

```



```

12763: Function wAddFontResource( FileName : PKOLChar ) : Integer
12764: //Function wAddFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector ) : Integer
12765: Function wCopyEnhMetaFile( p1 : HENHMETAFILE; p2 : PKOLChar ) : HENHMETAFILE
12766: Function wCopyMetaFile( p1 : HMETAFILE; p2 : PKOLChar ) : HMETAFILE
12767: //Function wCreateColorSpace( var ColorSpace : TLogColorSpace ) : HCOLORSPACE
12768: //Function wCreateDC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvInit : PDeviceMode ) : HDC
12769: //Function wCreateEnhMetaFile( DC : HDC; FileName : PKOLChar; Rect : PRect; Desc : PKOLChar ) : HDC
12770: Function wCreateFont( nHeight, nWidth, nEscapement, nOrientation, fnWeight : Integer; fdwItalic,
fdwUnderline, fdwStrikeOut, fdwCharSet, fdwOutputPrec, fdwClipPrecision, fdwQualy,
fdwPitchAndFamily : DWORD; lpszFace : PKOLChar ) : HFONT;
12771: Function wCreateFontIndirect( const p1 : TLogFont ) : HFONT
12772: //Function wCreateFontIndirectEx( const p1 : PEnumLogFontExDV ) : HFONT
12773: //Function wCreateIC( lpszDriver, lpszDevice, lpszOutput : PKOLChar; lpdvInit : PDeviceMode ) : HDC
12774: Function wCreateMetaFile( p1 : PKOLChar ) : HDC
12775: Function wCreateScalableFontResource( p1 : DWORD; p2, p3, p4 : PKOLChar ) : BOOL
12776: //Function wDeviceCapabilities( pDriverName, pDeviceName,
pPort : PKOLChar; iIdx : Integer; pOut : PKOLChar; DevMod : PDeviceMode ) : Integer;
12777: //Function wEnumFontFamilies( DC : HDC; p2 : PKOLChar; p3 : TFontEnumProc; p4 : LPARAM ) : BOOL
12778: //Function wEnumFontFamiliesEx( DC : HDC; var p2 : TLogFont; p3 : TFontEnumProc; p4 : LPARAM; p5 : DWORD ) : BOOL;
12779: //Function wEnumFonts( DC : HDC; lpszFace : PKOLChar; fntempProc : TFontEnumProc; lpszData : PKOLChar ) : Integer;
12780: //Function wEnumICMProfiles( DC : HDC; ICMProc : TFontEnumProc; p3 : LPARAM ) : Integer
12781: //Function wExtTextOut( DC : HDC; X,
Y : Integer; Options : Longint; Rect : PRect; Str : PKOLChar; Count : Longint; Dx : Integer ) : BOOL
12782: //Function wGetCharABCWidths( DC : HDC; FirstChar, LastChar : Integer; const ABCStructs ) : BOOL
12783: //Function wGetCharABCWidthsFloat( DC : HDC; FirstChar, LastChar : Integer; const ABCFloatStructs ) : BOOL
12784: //Function wGetCharWidth32( DC : HDC; FirstChar, LastChar : Integer; const Widths ) : BOOL
12785: //Function wGetCharWidth( DC : HDC; FirstChar, LastChar : Integer; const Widths ) : BOOL
12786: //Function wGetCharWidthFloat( DC : HDC; FirstChar, LastChar : Integer; const Widths ) : BOOL
12787: //Function wGetCharacterPlacement( DC : HDC; p2 : PKOLChar; p3, p4 : BOOL; var p5 : TGCPResults; p6 : DWORD ) : DWORD
12788: Function wGetEnhMetaFile( p1 : PKOLChar ) : HENHMETAFILE
12789: Function wGetEnhMetaFileDescription( p1 : HENHMETAFILE; p2 : Integer; p3 : PKOLChar ) : Integer
12790: //Function wGetGlyphIndices( DC : HDC; p2 : PKOLChar; p3 : Integer; p4 : PWORD; p5 : DWORD ) : DWORD
12791: //Function wGetGlyphOutline( DC : HDC; uChar, uFormat : Integer; const lpgm : TGlyphMetrics; cbBuffer : DWORD;
lpvBuffer : Pointer; const lpmat2 : TMat2 ) : DWORD
12792: Function wGetICMProfile( DC : HDC; var Size : DWORD; Name : PKOLChar ) : BOOL
12793: //Function wGetLogColorSpace( p1 : HCOLORSPACE; var ColorSpace : TLogColorSpace; Size : DWORD ) : BOOL
12794: Function wGetMetaFile( p1 : PKOLChar ) : HMETAFILE
12795: //Function wGetObject( p1 : HGDIOBJ; p2 : Integer; p3 : Pointer ) : Integer
12796: //Function wGetOutlineTextMetrics( DC : HDC; p2 : Integer; OTMetricStructs : Pointer ) : Integer
12797: //Function wGetTextExtentExPoint( DC : HDC; p2 : PKOLChar; p3, p4 : Integer; p5, p6 : Integer; var p7 : TSize ) : BOOL
12798: Function wGetTextExtentPoint32( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize ) : BOOL
12799: Function wGetTextExtentPoint( DC : HDC; Str : PKOLChar; Count : Integer; var Size : TSize ) : BOOL
12800: Function wGetTextFace( DC : HDC; Count : Integer; Buffer : PKOLChar ) : Integer
12801: //Function wGetTextMetrics( DC : HDC; var TM : TTextMetric ) : BOOL
12802: Function wPolyTextOut( DC : HDC; const PolyTextArray, Strings : Integer ) : BOOL
12803: Function wRemoveFontResource( FileName : PKOLChar ) : BOOL
12804: //Function wRemoveFontResourceEx( p1 : PKOLChar; p2 : DWORD; p3 : PDesignVector ) : BOOL
12805: //Function wResetDC( DC : HDC; const InitData : TDeviceMode ) : HDC
12806: Function wSetICMProfile( DC : HDC; Name : PKOLChar ) : BOOL
12807: //Function wStartDoc( DC : HDC; const p2 : TDocInfo ) : Integer
12808: Function wTextOut( DC : HDC; X, Y : Integer; Str : PKOLChar; Count : Integer ) : BOOL
12809: Function wUpdateICMRegKey( p1 : DWORD; p2, p3 : PKOLChar; p4 : Integer ) : BOOL
12810: Function wglUseFontBitmaps( DC : HDC; p2, p3, p4 : DWORD ) : BOOL
12811: //Function wglUseFontOutlines( p1 : HDC; p2, p3, p4 : DWORD; p5, p6 : Single; p7 : Integer; p8 : PGlyphMetricsFloat ) : BOOL
12812: Function wAppendMenu( hMenu : HMENU; uFlags, uIDNewItem : Integer; lpNewItem : PKOLChar ) : BOOL
12813: Function wCallMsgFilter( var lpMsg : TMsg; nCode : Integer ) : BOOL
12814: //Function
wCallWindowProc( lpPrevWndFunc : TFontEnumProc; hWnd : HWND; Msg : Integer; wParam : WPARAM; lParam : LPARAM ) : LRESULT
12815: //Function wChangeDisplaySettings( var lpDevMode : TDeviceMode; dwFlags : DWORD ) : Longint
12816: //Function wChangeDisplaySettingsEx( lpszDeviceName : PKOLChar; var lpDevMode : TDeviceMode; hWnd : HWND;
dwFlags : DWORD; lParam : Pointer ) : Longint
12817: Function wChangeMenu( hMenu : HMENU; cmd : Integer; lpszNewItem : PKOLChar; cmdInsert : Integer; flags : Integer ) : BOOL;
12818: Function wCharLower( lpsz : PKOLChar ) : PKOLChar
12819: Function wCharLowerBuff( lpsz : PKOLChar; cchLength : DWORD ) : DWORD
12820: Function wCharNext( lpsz : PKOLChar ) : PKOLChar
12821: //Function wCharNextEx( CodePage : Word; lpCurrentChar : LPCSTR; dwFlags : DWORD ) : LPSTR
12822: Function wCharPrev( lpszStart : PKOLChar; lpszCurrent : PKOLChar ) : PKOLChar
12823: //Function wCharPrevEx( CodePage : Word; lpStart, lpCurrentChar : LPCSTR; dwFlags : DWORD ) : LPSTR
12824: Function wCharToOem( lpszSrc : PKOLChar; lpszDst : PKOLChar ) : BOOL
12825: Function wCharToOemBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD ) : BOOL
12826: Function wCharUpper( lpsz : PKOLChar ) : PKOLChar
12827: Function wCharUpperBuff( lpsz : PKOLChar; cchLength : DWORD ) : DWORD
12828: Function wCopyAcceleratorTable( hAccelSrc : HACCEL; var lpAccelDst, cAccelEntries : Integer ) : Integer
12829: Function wCreateAcceleratorTable( var Accel, Count : Integer ) : HACCEL
12830: //Function wCreateDesktop( lpszDesktop,
lpszDevice : PKOLChar; pDevMode : PDeviceMode; dwFlags : DWORD; dwDesiredAccess : DWORD; lpsa : PSecurityAttribs ) : HDESK
12831: //Function wCreateDialogIndirectParam( hInstance : HINST; const lpTemplate : TDlgTemplate; hWndParent :
HWND; lpDialogFunc : TFontEnumProc; dwInitParam : LPARAM ) : HWND
12832: //Function wCreateDialogParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND;
lpDialogFunc : TFontEnumProc; dwInitParam : LPARAM ) : HWND
12833: Function wCreateMDIWindow( lpClassName, lpWindowName : PKOLChar; dwStyle : DWORD; X, Y, nWidth, nHeight : Integer;
hWndParent : HWND; hInstance : HINST; lParam : LPARAM ) : HWND
12834: //Function wCreateWindowEx( dwExStyle : DWORD; lpClassName : PKOLChar; lpWindowName : PKOLChar; dwStyle : DWORD; X, Y,
nWidth, nHeight : Integer; hWndParent : HWND; hMenu : HMENU; hInstance : HINST; lParam : Pointer ) : HWND
12835: //Function wCreateWindowStation( lpwinsta : PKOLChar; dwReserv,
dwDesiredAccess : DWORD; lpsa : PSecurityAttribs ) : HWINSTA;
12836: Function wDefDlgProc( hWnd : HWND; Msg : Integer; wParam : WPARAM; lParam : LPARAM ) : LRESULT
12837: Function wDefFrameProc( hWnd : HWND; hWndMDIClient : HWND; uMsg : Integer; wParam : WPARAM; lParam : LPARAM ) : LRESULT;
12838: Function wDefMDIChildProc( hWnd : HWND; uMsg : Integer; wParam : WPARAM; lParam : LPARAM ) : LRESULT;

```

```

12839: Function wDefWindowProc( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM): LRESULT
12840: //Function wDialogBoxIndirectParam( hInstance : HINST; const lpDialogTemplate : TDlgTemplate; hWndParent
: HWND; lpDialogFunc : TFNDlgProc; dwInitParam : LPARAM ) : Integer
12841: //Function wDialogBoxParam( hInstance : HINST; lpTemplateName : PKOLChar; hWndParent : HWND; lpDialogFunc
: TFNDlgProc; dwInitParam : LPARAM ) : Integer
12842: Function wDispatchMessage( const lpMsg : TMsg ) : Longint
12843: Function wDlglDirList( hDlg:HWND;lpPathSpec:PKOLChar;nIDListBox,
nIDStaticPath:Integer;uFileType:UINT):Integer;
12844: Function wDlglDirListComboBox( hDlg:HWND;lpPathSpec:PKOLChar;nIDComboBox,
nIDStaticPath:Integer;uFileType:UINT):Integer;
12845: Function wDlglDirSelectComboBoxEx( hDlg : HWND; lpString:PKOLChar;nCount,nIDComboBox:Integer): BOOL
12846: Function wDlglDirSelectEx( hDlg : HWND; lpString : PKOLChar; nCount, nIDListBox : Integer ) : BOOL
12847: //Function wDrawState(DC:HDC;Brush:HBRUSH;CBFunc:TFNDrawStateProc;lData:LPARA;wDat:WPARA;x,y,cx,
cy:Integer;Flags:UINT):BOOL;
12848: Function wDrawText( hDC:HDC;lpString:PKOLChar;nCount:Integer;var lpRect:TRect;uFormat:UINT):Integer;
12849: Function wFindWindow( lpClassName, lpWindowName : PKOLChar ) : HWND
12850: Function wFindWindowEx( Parent, Child : HWND; ClassName, WindowName : PKOLChar ) : HWND
12851: //Function wGetAltTabInfo(hwnd:HWND;iItem:Integer;var
pati:TAltTabInfo;pszItemText:PKOLChar;cchItemText:UINT):BOOL;
12852: // Function wGetClassInfo( hInstance : HINST; lpClassName : PKOLChar; var lpWndClass : TWndClass ) : BOOL
12853: //Function wGetClassInfoEx( Instance : HINST; Classname : PKOLChar; var WndClass : TWndClassEx ) : BOOL
12854: Function wGetClassLong( hWnd : HWND; nIndex : Integer ) : DWORD
12855: Function wGetClassName( hWnd : HWND; lpClassName : PKOLChar; nMaxCount : Integer ) : Integer
12856: Function wGetClipboardFormatName( format : UINT; lpszFormatName:PKOLChar;cchMaxCount:Integer):Integer;
12857: Function wGetDlglItemText( hDlg:HWND;nIDlglItem:Integer;lpString:PKOLChar;nMaxCount:Integer):UINT
12858: Function wGetKeyNameText( lParam : Longint; lpString : PKOLChar; nSize : Integer ) : Integer
12859: Function wGetKeyboardLayoutName( pwszKLID : PKOLChar ) : BOOL
12860: //Function wGetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; var p4 : TMenuItemInfo ) : BOOL
12861: Function wGetMenuString(hMenu:HMENU;uIDItem:UINT;lpString:PKOLChar;nMaxCount:Integer;uFlag:UINT):Integer;
12862: Function wGetMessage( var lpMsg : TMsg; hWnd : HWND; wParam: WPARAM; lParam: LPARAM ) : Integer
12863: Function wGetProp( hWnd : HWND; lpString : PKOLChar ) : THandle
12864: //Function wGetTabbedTextExtent( hDC : HDC; lpString : PKOLChar; nCount, nTabPositions : Integer; var
lpnTabStopPositions ) : DWORD
12865: //Function wGetObjectInformation(hObj:THandle;nIndex:Integer;pvInfo:Ptr;nLength:DWORD;var
lpnLengthNeed:DWORD)BOOL;
12866: Function wGetWindowLong( hWnd : HWND; nIndex : Integer ) : Longint
12867: Function wGetWindowModuleFileName( hWnd : HWND; pszFileName : PKOLChar; cchFileNameMax : UINT ) : UINT
12868: Function wGetWindowText( hWnd : HWND; lpString : PKOLChar; nMaxCount : Integer ) : Integer
12869: Function wGetWindowTextLength( hWnd : HWND ) : Integer
12870: //Function wGrayString( hDC:HDC;hBrush:HBRUSH;lpOutFunc:TFNGrayStrProc;lpDat:LPARA;nCnt,X,Y,nWidt,
nHeigt:Integer):BOOL;
12871: Function wInsertMenu( hMenu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar ) : BOOL
12872: //Function wInsertMenuItem( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo ) : BOOL
12873: Function wIsCharAlpha( ch : KOLChar ) : BOOL
12874: Function wIsCharAlphaNumeric( ch : KOLChar ) : BOOL
12875: Function wIsCharLower( ch : KOLChar ) : BOOL
12876: Function wIsCharUpper( ch : KOLChar ) : BOOL
12877: Function wIsDialogMessage( hDlg : HWND; var lpMsg : TMsg ) : BOOL
12878: Function wLoadAccelerators( hInstance : HINST; lpTableName : PKOLChar ) : HACCEL
12879: Function wLoadBitmap( hInstance : HINST; lpBitmapName : PKOLChar ) : HBITMAP
12880: Function wLoadCursor( hInstance : HINST; lpCursorName : PKOLChar ) : HCURSOR
12881: Function wLoadCursorFromFile( lpFileName : PKOLChar ) : HCURSOR
12882: Function wLoadIcon( hInstance : HINST; lpIconName : PKOLChar ) : HICON
12883: Function wLoadImage(hInst:HINST;ImageName:PKOLChar;ImageType:UINT;X,Y:Integer;Flags:UINT): THandle
12884: Function wLoadKeyboardLayout( pwszKLID : PKOLChar; Flags : UINT ) : HKL
12885: Function wLoadMenu( hInstance : HINST; lpMenuName : PKOLChar ) : HMENU
12886: //Function wLoadMenuIndirect( lpMenuTemplate : Pointer ) : HMENU
12887: Function wLoadString(hInstance:HINST;uID:UINT;lpBuffer:PKOLChar;nBufferMax:Integer):Integer
12888: Function wMapVirtualKey( uCode, uMapType : UINT ) : UINT
12889: Function wMapVirtualKeyEx( uCode, uMapType : UINT; dwHKL : HKL ) : UINT
12890: Function wMessageBox( hWnd : HWND; lpText, lpCaption : PKOLChar; uType : UINT ) : Integer
12891: Function wMessageBoxEx( hWnd:HWND; lpText,lpCaption:PKOLChar;uType:UINT;wLanguageId:Word): Integer
12892: //Function wMessageBoxIndirect( const MsgBoxParams : TMsgBoxParams ) : BOOL
12893: Function wModifyMenu( hMenu : HMENU; uPosition, uFlags, uIDNewItem : UINT; lpNewItem : PKOLChar ) : BOOL
12894: //Function wOemToAnsi( const lpszSrc : LPCSTR; lpszDst : LPSTR ) : BOOL
12895: //7Function wOemToAnsiBuff( lpszSrc : LPCSTR; lpszDst : LPSTR; cchDstLength : DWORD ) : BOOL
12896: //Function wOemToChar( lpszSrc : PKOLChar; lpszDst : PKOLChar ) : BOOL
12897: Function wOemToCharBuff( lpszSrc : PKOLChar; lpszDst : PKOLChar; cchDstLength : DWORD ) : BOOL
12898: Function wOpenDesktop(lpszDesktop:PKOLChar;dwFlags:DWORD;fInherit:BOOL;dwDesiredAccess:DWORD): HDESK
12899: Function wOpenWindowStation( lpszWinSta : PKOLChar; fInherit : BOOL; dwDesiredAccess : DWORD ) : HWINSTA
12900: Function wPeekMessage( var lpMsg : TMsg; hWnd : HWND; wParam: WPARAM; lParam: LPARAM ) : Integer
12901: Function wPostMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
12902: Function wPostThreadMessage(idThread:DWORD;Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
12903: Function wRealGetWindowClass( hWnd : HWND; pszType : PKOLChar; cchType : UINT ) : UINT
12904: // Function wRegisterClass( const lpWndClass : TWndClass ) : ATOM
12905: // Function wRegisterClassEx( const WndClass : TWndClassEx ) : ATOM
12906: Function wRegisterClipboardFormat( lpszFormat : PKOLChar ) : UINT
12907: // Function wRegisterDeviceNotification(hRecipient:THandle;NotificFilter:Pointer;Flags:DWORD):HDEVNOTIFY
12908: Function wRegisterWindowMessage( lpString : PKOLChar ) : UINT
12909: Function wRemoveProp( hWnd : HWND; lpString : PKOLChar ) : THandle
12910: Function wSendDlglItemMessage(hDlg:HWND;nIDlglItem:Integer;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
12911: Function wSendMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : LRESULT
12912: //Function wSendMessageCallback( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam:LPARAM;
lpResultCallBack : TFNSendAsyncProc; dwData : DWORD ) : BOOL
12913: Function wSendMessageTimeout(hWnd:HWND;Msg:UINT;wParam:WPARAM;lParam:LPARAM;fuFlags,uTimeout:UINT;var
lpdwResult:DWORD): LRESULT
12914: Function wSendNotifyMessage( hWnd : HWND; Msg : UINT; wParam : WPARAM; lParam : LPARAM ) : BOOL
12915: Function wSetClassLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint ) : DWORD
12916: Function wSetDlglItemText( hDlg : HWND; nIDlglItem : Integer; lpString : PKOLChar ) : BOOL

```

```

12917: //Function wSetMenuItemInfo( p1 : HMENU; p2 : UINT; p3 : BOOL; const p4 : TMenuItemInfo) : BOOL
12918: Function wSetProp( hWnd : HWND; lpString : PKOLChar; hData : THandle) : BOOL
12919: // Function wSetUserObjectInformation(hObj:THandle;nIndex:Integer;pvInfo:Pointer;nLength:DWORD):BOOL
12920: Function wSetWindowLong( hWnd : HWND; nIndex : Integer; dwNewLong : Longint) : Longint
12921: Function wSetWindowText( hWnd : HWND; lpString : PKOLChar) : BOOL
12922: //Function wSetWindowsHook( nFilterType : Integer; pfnFilterProc : TFNHookProc) : HHOOK
12923: //Function wSetWindowsHookEx(idHook:Integer;lpfn:TFNHookProc;hmod:HINST;dwThreadId:DWORD):HHOOK;
12924: // Function wSystemParametersInfo( uiAction, uiParam : UINT; pvParam : Pointer; fWinIni: UINT):BOOL
12925: Function wTabbedTextOut(hDC:HDC;X,Y:Int;lpString:PKOLChar;nCount,nTabPositions:Int;var
    lpnTabStopPositions,nTabOrigin:Int):Longint;
12926: Function wTranslateAccelerator( hWnd : HWND; hAccTable : HACCEL; var lpMsg : TMsg) : Integer
12927: Function wUnregisterClass( lpClassName : PKOLChar; hInstance : HINST) : BOOL
12928: Function wVkKeyScan( ch : KOLChar) : SHORT
12929: Function wVkKeyScanEx( ch : KOLChar; dwHKL : HKL) : SHORT
12930: Function wWinHelp( hWndMain : HWND; lpszHelp : PKOLChar; uCommand : UINT; dwData : DWORD) : BOOL
12931: Function wwsprintf( Output : PKOLChar; Format : PKOLChar) : Integer
12932: Function wwvsprintf( Output : PKOLChar; Format : PKOLChar; arglist : va_list) : Integer
12933:
12934: //TestDrive!
12935: 'SID_REVISION', 'LongInt').SetInt(1); 'FILENAME_ADVAPI32', 'String').SetString('ADVAPI32.DLL
12936: 'PROC_CONVERTSIDTOSTRINGSIDA', 'String').SetString('ConvertSidToStringSida
12937: Function GetDomainUserSidS(const domainName:String;const userName:String; var foundDomain:String):String;
12938: Function GetLocalUserSidStr( const UserName : string) : string
12939: Function getPid4user( const domain : string; const user : string; var pid : dword) : boolean
12940: Function impersonate2User( const domain : string; const user : string) : boolean
12941: Function GetProcessUserByPid( pid : DWORD; var UserName, Domain : AnsiString) : Boolean
12942: Function KillProcessbyname( const exename : string; var found : integer) : integer
12943: Function getWinProcessList : TStringList
12944: end;
12945:
12946: procedure SIRegister_AfSafeSync(CL: TPSPascalCompiler);
12947: begin
12948: 'AfMaxSyncSlots', 'LongInt').SetInt( 64);
12949: 'AfSynchronizeTimeout', 'LongInt').SetInt( 2000);
12950: 'TafSyncSlotID', 'DWORD
12951: 'TafSyncStatistics', 'record MessagesCount:Int;TimeoutMessages:Int;DisabledMessages:Int;end;
12952: 'TafSafeSyncEvent', 'Procedure ( ID : TafSyncSlotID)
12953: 'TafSafeDirectSyncEvent', 'Procedure
12954: Function AfNewSyncSlot( const AEvent : TafSafeSyncEvent) : TafSyncSlotID
12955: Function AfReleaseSyncSlot( const ID : TafSyncSlotID) : Boolean
12956: Function AfEnableSyncSlot( const ID : TafSyncSlotID; Enable : Boolean) : Boolean
12957: Function AfValidateSyncSlot( const ID : TafSyncSlotID) : Boolean
12958: Function AfSyncEvent( const ID : TafSyncSlotID; Timeout : DWORD) : Boolean
12959: Function AfDirectSyncEvent( Event : TafSafeDirectSyncEvent; Timeout : DWORD) : Boolean
12960: Function AfIsSyncMethod : Boolean
12961: Function AfSyncWnd : HWND
12962: Function AfSyncStatistics : TafSyncStatistics
12963: Procedure AfClearSyncStatistics
12964: end;
12965:
12966: procedure SIRegister_AfComPortCore(CL: TPSPascalCompiler);
12967: begin
12968: 'fBinary', 'LongWord').SetUInt( $00000001);
12969: 'fParity', 'LongWord').SetUInt( $00000002);
12970: 'fOutxCtsFlow', 'LongWord').SetUInt( $00000004);
12971: 'fOutxDsrFlow', 'LongWord').SetUInt( $00000008);
12972: 'fDtrControl', 'LongWord').SetUInt( $00000030);
12973: 'fDtrControlDisable', 'LongWord').SetUInt( $00000000);
12974: 'fDtrControlEnable', 'LongWord').SetUInt( $00000010);
12975: 'fDtrControlHandshake', 'LongWord').SetUInt( $00000020);
12976: 'fDsrSensitivity', 'LongWord').SetUInt( $00000040);
12977: 'fTXContinueOnXoff', 'LongWord').SetUInt( $00000080);
12978: 'fOutX', 'LongWord').SetUInt( $00000100);
12979: 'fInX', 'LongWord').SetUInt( $00000200);
12980: 'fErrorChar', 'LongWord').SetUInt( $00000400);
12981: 'fNull', 'LongWord').SetUInt( $00000800);
12982: 'fRtsControl', 'LongWord').SetUInt( $00003000);
12983: 'fRtsControlDisable', 'LongWord').SetUInt( $00000000);
12984: 'fRtsControlEnable', 'LongWord').SetUInt( $00001000);
12985: 'fRtsControlHandshake', 'LongWord').SetUInt( $00002000);
12986: 'fRtsControlToggle', 'LongWord').SetUInt( $00003000);
12987: 'fAbortOnError', 'LongWord').SetUInt( $00004000);
12988: 'fDummy2', 'LongWord').SetUInt( $FFFF8000);
12989: 'TafCoreEvent', '( ceOutFree, ceLineEvent, ceNeedReadData, ceException )
12990: FindClass('TOBJECT'), 'EAfComPortCoreError
12991: FindClass('TOBJECT'), 'TafComPortCore
12992: 'TafComPortCoreEvent', 'Procedure ( Sender : TafComPortCore; Even'
12993: + 'tKind : TafCoreEvent; Data : DWORD)
12994: SIRegister_TafComPortCoreThread(CL);
12995: SIRegister_TafComPortEventThread(CL);
12996: SIRegister_TafComPortWriteThread(CL);
12997: SIRegister_TafComPortCore(CL);
12998: Function FormatDeviceName( PortNumber : Integer) : string
12999: end;
13000:
13001: procedure SIRegister_ApplicationFileIO(CL: TPSPascalCompiler);
13002: begin
13003: 'TAPIOFileStreamEvent', 'Function ( const fileName : String; mode: Word) : TStream
13004: 'TAPIOFileStreamExistsEvent', 'Function ( const fileName : String) : Boolean

```

```

13005: SIRegister_TApplicationFileIO(CL);
13006: TDataFileCapability', '( dfcRead, dfcWrite )
13007: TDataFileCapabilities', 'set of TDataFileCapability
13008: SIRegister_TDataFile(CL);
13009: //TDataFileClass', 'class of TDataFile
13010: Function ApplicationFileIODefined : Boolean
13011: Function CreateFileStream(const fileName: String; mode: WordfmShareDenyNone):TStream
13012: Function FileStreamExists(const fileName: String) : Boolean
13013: //Procedure Register
13014: end;
13015:
13016: procedure SIRegister_ALFBXLib(CL: TPSPascalCompiler);
13017: begin
13018:   TALFBXFieldType', '( uftUnknown, uftNumeric, uftChar, uftVarchar'
13019:   +', uftCString, uftSmallint, uftInteger, uftQuad, uftFloat, uftDoublePrecisi'
13020:   +'on, uftTimestamp, uftBlob, uftBlobId, uftDate, uftTime, uftInt64, uftArray, uftNull )
13021:   TALFBXScale', 'Integer
13022:   FindClass('TOBJECT'),'EALFBXConvertError
13023:   SIRegister_EALFBXError(CL);
13024:   SIRegister_EALFBXException(CL);
13025:   FindClass('TOBJECT'),'EALFBXGfixError
13026:   FindClass('TOBJECT'),'EALFBXDSQLError
13027:   FindClass('TOBJECT'),'EALFBXDynError
13028:   FindClass('TOBJECT'),'EALFBXGBakError
13029:   FindClass('TOBJECT'),'EALFBXGSecError
13030:   FindClass('TOBJECT'),'EALFBXLicenseError
13031:   FindClass('TOBJECT'),'EALFBXGStatError
13032:   //EALFBXExceptionClass', 'class of EALFBXError
13033:   TALFBXCharacterSet', '( csNONE, csASCII, csBIG_5, csCYRL, csDOS4'
13034:   +'37, csDOS850, csDOS852, csDOS857, csDOS860, csDOS861, csDOS863, csDOS865, '
13035:   +'csEUCJ_0208, csGB_2312, csISO8859_1, csISO8859_2, csKSC_5601, csNEXT, csOC'
13036:   +'TETS, csSJIS_0208, csUNICODE_FSS, csUTF8, csWIN1250, csWIN1251, csWIN1252,'
13037:   +'csWIN1253, csWIN1254, csDOS737, csDOS775, csDOS858, csDOS862, csDOS864, c'
13038:   +'sDOS866, csDOS869, csWIN1255, csWIN1256, csWIN1257, csISO8859_3, csISO8859'
13039:   +'_4, csISO8859_5, csISO8859_6, csISO8859_7, csISO8859_8, csISO8859_9, csISO'
13040:   +'8859_13, csKOI8R, csKOI8U, csWIN1258, csTIS620, csGBK, csCP943C )
13041:   TALFBXTransParam', '( tpConsistency, tpConcurrency, tpShared, tp'
13042:   +'Protected, tpExclusive, tpWait, tpNowait, tpRead, tpWrite, tpLockRead, tpL'
13043:   +'ockWrite, tpVerbTime, tpCommitTime, tpIgnoreLimbo, tpReadCommitted, tpAuto'
13044:   +'Commit, tpRecVersion, tpNoRecVersion, tpRestartRequests, tpNoAutoUndo, tpLockTimeout )
13045:   TALFBXTransParams', 'set of TALFBXTransParam
13046:   Function ALFBXStrToCharacterSet( const CharacterSet : AnsiString) : TALFBXCharacterSet
13047:   Function ALFBXCreateDBParams( Params : AnsiString; Delimiter : Char) : AnsiString
13048:   Function ALFBXCreateBlobParams( Params : AnsiString; Delimiter : Char) : AnsiString
13049:   'cALFBXMaxParamLength', 'LongInt').SetInt( 125);
13050:   TALFBXParamsFlag', '( pfNotInitialized, pfNotNullable )
13051:   TALFBXParamsFlags', 'set of TALFBXParamsFlag
13052:   //PALFBXSQLVar', '^TALFBXSQLVar // will not work
13053:   //PALFBXSQLDaData', '^TALFBXSQLDaData // will not work
13054:   TALFBXStatementType', '( stSelect, stInsert, stUpdate, stDelete,'
13055:   +' stDDL, stGetSegment, stPutSegment, stExecProcedure, stStartTrans, stCommi'
13056:   +'t, stRollback, stSelectForUpdate, stSetGenerator, stSavePoint )
13057:   SIRegister_TALFBXSQLDA(CL);
13058:   //PALFBXPtrArray', '^TALFBXPtrArray // will not work
13059:   SIRegister_TALFBXPoolStream(CL);
13060:   //PALFBXBlobData', '^TALFBXBlobData // will not work
13061:   TALFBXBlobData', 'record Size : Integer; Buffer : string; end
13062:   //PALFBXArrayDesc', '^TALFBXArrayDesc // will not work
13063:   //TALFBXArrayDesc', 'TISCArrDesc
13064:   //TALFBXBlobDesc', 'TISCBlobDesc
13065:   //PALFBXArrayInfo', '^TALFBXArrayInfo // will not work
13066:   //TALFBXArrayInfo', 'record index : Integer; size : integer; info: TALFBXArrayDesc; end
13067:   SIRegister_TALFBXSQLResult(CL);
13068:   //TALFBXSQLResultClass', 'class of TALFBXSQLResult
13069:   SIRegister_TALFBXSQLParams(CL);
13070:   //TALFBXSQLParamsClass', 'class of TALFBXSQLParams
13071:   TALFBXDSQLInfoData', 'record InfoCode : byte; InfoLen : Word; St'
13072:   +'atementType : TALFBXStatementType; end
13073:   FindClass('TOBJECT'),'TALFBXLibrary
13074:   //PALFBXStatusVector', '^TALFBXStatusVector // will not work
13075:   TALFBXOnConnectionLost', 'Procedure ( Lib : TALFBXLibrary)
13076:   //TALFBXOnGetDBExceptionClass', 'Procedure ( Number : Integer; out'
13077:   //'+' Excep : EALFBXExceptionClass)
13078:   SIRegister_TALFBXLibrary(CL);
13079:   'cALFBXDateOffset', 'LongInt').SetInt( 15018);
13080:   'cALFBXTimeCoeff', 'LongInt').SetInt( 864000000);
13081:   //Procedure ALFBXDecodeTimeStamp( v : PISCTimeStamp; out DateTime : Double);
13082:   //Procedure ALFBXDecodeTimeStamp1( v : PISCTimeStamp; out TimeStamp : TTimeStamp);
13083:   //Function ALFBXDecodeTimeStamp2( v : PISCTimeStamp) : Double;
13084:   Procedure ALFBXDecodeSQLDate( v : Integer; out Year : SmallInt; out Month, Day : Word)
13085:   Procedure ALFBXDecodeSQLTime(v: Cardinal; out Hour, Minute, Second: Word; out Fractions: LongWord)
13086:   //Procedure ALFBXEncodeTimeStamp( const DateTime : TDateTime; v : PISCTimeStamp);
13087:   //Procedure ALFBXEncodeTimeStamp1( const Date : Integer; v : PISCTimeStamp);
13088:   //Procedure ALFBXEncodeTimeStamp2( const Time : Cardinal; v : PISCTimeStamp);
13089:   Function ALFBXEncodeSQLDate( Year : Integer; Month, Day : Integer) : Integer
13090:   Function ALFBXEncodeSQLTime( Hour, Minute, Second : Word; var Fractions : LongWord): Cardinal
13091:   TALFBXParamType', '( prNone, prByte, prShrt, prCard, prStrg, prIgno )
13092:   TALFBXDPBInfo', 'record Name : AnsiString; ParamType : TALFBXParamType; end
13093:   Function ALFBXSQLQuote( const name : AnsiString) : AnsiString

```



```

13094: Function ALFBXSQLUnQuote( const name : AnsiString) : AnsiString
13095: end;
13096:
13097: procedure SIRegister_ALFBXClient(CL: TPSPascalCompiler);
13098: begin
13099:   TALFBXClientSQLParam', 'record Value : AnsiString; IsNull : Boolean; end
13100:   TALFBXClientSQLParams', 'array of TALFBXClientSQLParam
13101:   TALFBXClientSelectDataSQL', 'record SQL : AnsiString; Params : T'
13102:   + 'ALFBXClientSQLParams; RowTag : AnsiString; ViewTag : AnsiString; Skip : in'
13103:   + 'teger; First : Integer; CacheThreshold : Integer; end
13104:   TALFBXClientSelectDataSQLs', 'array of TALFBXClientSelectDataSQL
13105:   TALFBXClientUpdateDataSQL', 'record SQL : AnsiString; Params: TALFBXClientSQLParams; end
13106:   TALFBXClientUpdateDataSQLs', 'array of TALFBXClientUpdateDataSQL
13107:   TALFBXClientMonitoringIOStats', 'record page_reads : int64; page'
13108:   + '_writes : int64; page_fetches : int64; page_marks : int64; end
13109:   SIRegister_TALFBXClient(CL);
13110:   SIRegister_TALFBXConnectionStatementPoolBinTreeNode(CL);
13111:   SIRegister_TALFBXConnectionStatementPoolBinTree(CL);
13112:   SIRegister_TALFBXConnectionWithStmtPoolContainer(CL);
13113:   SIRegister_TALFBXConnectionWithoutStmtPoolContainer(CL);
13114:   SIRegister_TALFBXReadTransactionPoolContainer(CL);
13115:   SIRegister_TALFBXReadStatementPoolContainer(CL);
13116:   SIRegister_TALFBXStringKeyPoolBinTreeNode(CL);
13117:   SIRegister_TALFBXConnectionPoolClient(CL);
13118:   SIRegister_TALFBXEventThread(CL);
13119:   Function AlMySqlClientSlashedStr( const Str : AnsiString) : AnsiString
13120: end;
13121:
13122: procedure SIRegister_ovcBidi(CL: TPSPascalCompiler);
13123: begin
13124:   _OSVERSIONINFOA = record
13125:     dwOSVersionInfoSize: DWORD;
13126:     dwMajorVersion: DWORD;
13127:     dwMinorVersion: DWORD;
13128:     dwBuildNumber: DWORD;
13129:     dwPlatformId: DWORD;
13130:     szCSDVersion: array[0..127] of AnsiChar; { Maintenance AnsiString for PSS usage }
13131:   end;
13132:   TOSVersionInfoA', '_OSVERSIONINFOA
13133:   TOSVersionInfo', 'TOSVersionInfoA
13134:   'WS_EX_RIGHT', 'LongWord').SetUInt( $00001000);
13135:   'WS_EX_LEFT', 'LongWord').SetUInt( $00000000);
13136:   'WS_EX_RTLREADING', 'LongWord').SetUInt( $00002000);
13137:   'WS_EX_LTRREADING', 'LongWord').SetUInt( $00000000);
13138:   'WS_EX_LEFTSCROLLBAR', 'LongWord').SetUInt( $00004000);
13139:   'WS_EX_RIGHTSCROLLBAR', 'LongWord').SetUInt( $00000000);
13140:   Function SetProcessDefaultLayout( dwDefaultLayout : DWORD) : BOOL
13141:   'LAYOUT_RTL', 'LongWord').SetUInt( $00000001);
13142:   'LAYOUT_BTT', 'LongWord').SetUInt( $00000002);
13143:   'LAYOUT_VBH', 'LongWord').SetUInt( $00000004);
13144:   'LAYOUT_BITMAPORIENTATIONPRESERVED', 'LongWord').SetUInt( $00000008);
13145:   'NOMIRRORBITMAP', 'LongWord').SetUInt( DWORD ( $80000000 ));
13146:   Function SetLayout( dc : HDC; dwLayout : DWORD) : DWORD
13147:   Function GetLayout( dc : hdc) : DWORD
13148:   Function IsBidi : Boolean
13149:   Function GetCurrentHwProfile( var lpHwProfileInfo : THWProfileInfo) : BOOL
13150:   Function GetVersionEx( var lpVersionInformation : TOSVersionInfo) : BOOL
13151:   Function SetPriorityClass( hProcess : THandle; dwPriorityClass: DWORD) : BOOL
13152:   Function GetPriorityClass( hProcess : THandle) : DWORD
13153:   Function OpenClipboard( hWndNewOwner : HWND) : BOOL
13154:   Function CloseClipboard : BOOL
13155:   Function GetClipboardSequenceNumber : DWORD
13156:   Function GetClipboardOwner : HWND
13157:   Function SetClipboardViewer( hWndNewViewer : HWND) : HWND
13158:   Function GetClipboardViewer : HWND
13159:   Function ChangeClipboardChain( hWndRemove, hWndNewNext : HWND) : BOOL
13160:   Function SetClipboardData( uFormat : UINT; hMem : THandle) : THandle
13161:   Function GetClipboardData( uFormat : UINT) : THandle
13162:   Function RegisterClipboardFormat( lpszFormat : PChar) : UINT
13163:   Function CountClipboardFormats : Integer
13164:   Function EnumClipboardFormats( format : UINT) : UINT
13165:   Function GetClipboardFormatName( format:UINT;lpszFormatName:PChar;cchMaxCount:Integer):Integer
13166:   Function EmptyClipboard : BOOL
13167:   Function IsClipboardFormatAvailable( format : UINT) : BOOL
13168:   Function GetPriorityClipboardFormat( var paFormatPriorityList, cFormats : Integer) : Integer
13169:   Function GetOpenClipboardWindow : HWND
13170:   Function EndDialog( hDlg : HWND; nResult : Integer) : BOOL
13171:   Function GetDlgItem( hDlg : HWND; nIDDlgItem : Integer) : HWND
13172:   Function SetDlgItemInt( hDlg : HWND; nIDDlgItem : Integer; uValue : UINT; bSigned: BOOL): BOOL
13173:   Function GetDlgItemInt(hDlg:HWND;nIDDlgItem:Integer;var lpTranslated:BOOL;bSigned: BOOL): UINT
13174:   Function SetDlgItemText( hDlg : HWND; nIDDlgItem : Integer; lpString : PChar) : BOOL
13175:   Function CheckDlgButton( hDlg : HWND; nIDButton : Integer; uCheck : UINT) : BOOL
13176:   Function CheckRadioButton( hDlg : HWND; nIDFirstButton, nIDLastButton, nIDCheckButton : Integer) : BOOL
13177:   Function IsDlgButtonChecked( hDlg : HWND; nIDButton : Integer) : UINT
13178:   Function SendDlgItemMessage(hDlg:HWND;nIDDlgItem: Int;Msg:UINT;wParam:WPARAM;lParam:LPARAM):Longint;
13179: end;
13180:
13181: procedure SIRegister_DXPUtils(CL: TPSPascalCompiler);
13182: begin

```

```

13183: Function glExecuteAndWait(cmdLine:String;visibility:Word;timeout:Cardinal;killAppOnTimeOut:Bool):Int;
13184: Function GetTemporaryFilesPath : String
13185: Function GetTemporaryFileName : String
13186: Function FindFileInPaths( const fileName, paths : String ) : String
13187: Function PathsToString( const paths : TStrings ) : String
13188: Procedure StringToPaths( const pathsString : String; paths : TStrings)
13189: //Function MacroExpandPath( const aPath : String ) : String
13190: end;
13191:
13192: procedure SIRegister_ALMultiPartBaseParser(CL: TPSPascalCompiler);
13193: begin
13194:   SIRegister_TALMultiPartBaseContent(CL);
13195:   SIRegister_TALMultiPartBaseContents(CL);
13196:   SIRegister_TALMultiPartBaseStream(CL);
13197:   SIRegister_TALMultiPartBaseEncoder(CL);
13198:   SIRegister_TALMultiPartBaseDecoder(CL);
13199:   Function ALMultiPartExtractBoundaryFromContentType( aContentType : AnsiString ) : AnsiString
13200:   Function ALMultiPartExtractSubValueFromHeaderLine(aHeaderLine:AnsiString;aName:AnsiString):AnsiString;
13201:   Function ALMultiPartSetSubValueInHeaderLine(aHeaderLine:AnsiString;aName,AValue:AnsiString):AnsiString;
13202: end;
13203:
13204: procedure SIRegister_SmallUtils(CL: TPSPascalCompiler);
13205: begin
13206:   TDriveSize', 'record FreeS : Int64; Totals : Int64; end
13207:   TWinVerRec', 'record WinPlatform : Integer; WinMajorVersion : Int'
13208:   + 'eger; WinMinorVersion : Integer; WinBuildNumber : Integer; WinCSDVersion: String; end
13209:   Function aAllocPadedMem( Size : Cardinal ) : TObject
13210:   Procedure aFreePadedMem( var P : TObject);
13211:   Procedure aFreePadedMeml( var P : PChar);
13212:   Function aCheckPadedMem( P : Pointer ) : Byte
13213:   Function aGetPadMemSize( P : Pointer ) : Cardinal
13214:   Function aAllocMem( Size : Cardinal ) : Pointer
13215:   Function aStrLen( const Str : PChar ) : Cardinal
13216:   Function aStrLCopy( Dest : PChar; const Source : PChar; MaxLen : Cardinal ) : PChar
13217:   Function aStrECopy( Dest : PChar; const Source : PChar ) : PChar
13218:   Function aStrCopy( Dest : PChar; const Source : PChar ) : PChar
13219:   Function aStrEnd( const Str : PChar ) : PChar
13220:   Function aStrScan( const Str : PChar; aChr : Char ) : PChar
13221:   Function aStrMove( Dest : PChar; const Source : PChar; Count : Cardinal ) : PChar
13222:   Function aPCharLength( const Str : PChar ) : Cardinal
13223:   Function aPCharUpper( Str : PChar ) : PChar
13224:   Function aPCharLower( Str : PChar ) : PChar
13225:   Function aStrCat( Dest : PChar; const Source : PChar ) : PChar
13226:   Function aLastDelimiter( const Delimiters, S : String ) : Integer
13227:   Function aCopyTail( const S : String; Len : Integer ) : String
13228:   Function aInt2Thos( I : Int64 ) : String
13229:   Function aUpperCase( const S : String ) : String
13230:   Function aLowerCase( const S : string ) : String
13231:   Function aCompareText( const S1, S2 : string ) : Integer
13232:   Function aSameText( const S1, S2 : string ) : Boolean
13233:   Function aInt2Str( Value : Int64 ) : String
13234:   Function aStr2Int( const Value : String ) : Int64
13235:   Function aStr2IntDef( const S : string; Default : Int64 ) : Int64
13236:   Function aGetFileExt( const FileName : String ) : String
13237:   Function aGetFilePath( const FileName : String ) : String
13238:   Function aGetFileName( const FileName : String ) : String
13239:   Function aChangeExt( const FileName, Extension : String ) : String
13240:   Function aAdjustLineBreaks( const S : string ) : string
13241:   Function aGetWindowStr( WinHandle : HWND ) : String
13242:   Function aDiskSpace( Drive : String ) : TDriveSize
13243:   Function aFileExists( FileName : String ) : Boolean
13244:   Function aFileSize( FileName : String ) : Int64
13245:   Function aDirectoryExists( const Name : string ) : Boolean
13246:   Function aSysErrorMessage( ErrorCode : Integer ) : string
13247:   Function aShortPathName( const LongName : string ) : string
13248:   Function aGetWindowVer : TWinVerRec
13249:   procedure InitDriveSpacePtr;
13250: end;
13251:
13252: procedure SIRegister_MakeApp(CL: TPSPascalCompiler);
13253: begin
13254:   aZero', 'LongInt').SetInt( 0 );
13255:   'makeappDEF', 'LongInt').SetInt( - 1 );
13256:   'CS_VREDRAW', 'LongInt').SetInt( DWORD ( 1 ));
13257:   'CS_HREDRAW', 'LongInt').SetInt( DWORD ( 2 ));
13258:   'CS_KEYCVTWINDOW', 'LongInt').SetInt( 4 );
13259:   'CS_DBLCLKS', 'LongInt').SetInt( 8 );
13260:   'CS_OWNDC', 'LongWord').SetUInt( $20 );
13261:   'CS_CLASSDC', 'LongWord').SetUInt( $40 );
13262:   'CS_PARENTDC', 'LongWord').SetUInt( $80 );
13263:   'CS_NOKEYCVT', 'LongWord').SetUInt( $100 );
13264:   'CS_NOCLOSE', 'LongWord').SetUInt( $200 );
13265:   'CS_SAVEBITS', 'LongWord').SetUInt( $800 );
13266:   'CS_BYTEALIGNCLIENT', 'LongWord').SetUInt( $1000 );
13267:   'CS_BYTEALIGNWINDOW', 'LongWord').SetUInt( $2000 );
13268:   'CS_GLOBALCLASS', 'LongWord').SetUInt( $4000 );
13269:   'CS_IME', 'LongWord').SetUInt( $10000 );
13270:   'CS_DROPShadow', 'LongWord').SetUInt( $20000 );
13271:   //PPanelFunc', '^TPanelFunc // will not work

```

```

13272: TPanelStyle', '(psEdge, psTabEdge, psBorder, psTabBorder, psTab, psNone )
13273: TFontLook', '( flBold, flItalic, flUnderLine, flStrikeOut )
13274: TFontLooks', 'set of TFontLook
13275: TMessagefunc', 'function(hWnd,iMsg,wParam,lParam:Integer):Integer)
13276: Function SetWinClass(const ClassName:String; pMessFunc: Tmessagefunc; wcStyle : Integer): Word
13277: Function SetWinClassO( const ClassName : String; pMessFunc : TObject; wcStyle : Integer): Word
13278: Function SetWinClass( const ClassName : String; pMessFunc : TObject; wcStyle : Integer): Word
13279: Function MakeForm(Left,Top,Width,Height:Integer;const Caption:String;WinStyle:Integer):Integer
13280: Procedure RunMsgLoop( Show : Boolean)
13281: Function MakeFont(Height,Width:Integer; const FontName:String; Look:TFontLooks; Roman:Boolean): Integer
13282: Function MakeButton(Left,Top,Width,Height:Integer;pCaption:PChar;hParent,
ID_Number:Cardinal;hFont:Int):Int;
13283: Function MakeListBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Integer
13284: Function MakeComboBox(Left,Top,Width,Height,Parent:Integer;const ListItems:String;WinStyle:Integer):Int
13285: Function MakePanel(Left,Top,Width,Height,
hParent:Int;WndFunc:TPanelFunc;ID_Number:Card;Style:TPanelStyle):Int;
13286: Function MakeSubMenu(const ItemList : String; ID1, ID2 : Cardinal; hMenu : Integer) : Integer
13287: Function id4menu( a, b : Byte; c : Byte; d : Byte) : Cardinal
13288: Procedure DoInitMakeApp //set first to init formclasscontrol!
13289: end;
13290:
13291: procedure SIRegister_ScreenSaver(CL: TPSPascalCompiler);
13292: begin
13293:   TScreenSaverOption', '( ssoAutoAdjustFormProperties, ssoAutoHook'
13294:   + 'KeyboardEvents, ssoAutoHookMouseEvents, ssoEnhancedMouseMoveDetection )
13295:   TScreenSaverOptions', 'set of TScreenSaverOption
13296:   'cDefaultScreenSaverOptions', 'LongInt').Value.ts32:=ord(ssoAutoAdjustFormProperties) or
ord(ssoAutoHookKeyboardEvents) or ord(ssoEnhancedMouseMoveDetection);
13297:   TScreenSaverPreviewEvent', 'Procedure ( Sender : TObject; previewHwnd: HWND)
13298:   SIRegister_TScreenSaver(CL);
13299:   //Procedure Register
13300:   Procedure SetScreenSaverPassword
13301: end;
13302:
13303: procedure SIRegister_XCollection(CL: TPSPascalCompiler);
13304: begin
13305:   FindClass('TOBJECT'),'TXCollection
13306:   SIRegister_EFilerException(CL);
13307:   SIRegister_TXCollectionItem(CL);
13308:   //TXCollectionItemClass', 'class of TXCollectionItem
13309:   SIRegister_TXCollection(CL);
13310:   Procedure RegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13311:   Procedure DeRegisterXCollectionDestroyEvent( notifyEvent : TNotifyEvent)
13312:   Procedure RegisterXCollectionItemClass( aClass : TXCollectionItemClass)
13313:   Procedure UnregisterXCollectionItemClass( aClass : TXCollectionItemClass)
13314:   Function FindXCollectionItemClass( const className : String) : TXCollectionItemClass
13315:   Function GetXCollectionItemClassesList( baseClass : TXCollectionItemClass) : TList
13316: end;
13317:
13318: procedure SIRegister_XOpenGL(CL: TPSPascalCompiler);
13319: begin
13320:   TMapTexCoordMode', '( mtcUndefined, mtcNull, mtcMain, mtcDual, mtcSecond,mtcmArbitrary);
13321:   Procedure xglMapTexCoordToNull
13322:   Procedure xglMapTexCoordToMain
13323:   Procedure xglMapTexCoordToSecond
13324:   Procedure xglMapTexCoordToDual
13325:   Procedure xglMapTexCoordToArbitrary( const units : array of Cardinal);
13326:   Procedure xglMapTexCoordToArbitrary1( const bitWiseUnits : Cardinal);
13327:   Procedure xglMapTexCoordToArbitraryAdd( const bitWiseUnits : Cardinal)
13328:   Procedure xglBeginUpdate
13329:   Procedure xglEndUpdate
13330:   Procedure xglPushState
13331:   Procedure xglPopState
13332:   Procedure xglForbidSecondTextureUnit
13333:   Procedure xglAllowSecondTextureUnit
13334:   Function xglGetBitWiseMapping : Cardinal
13335: end;
13336:
13337: procedure SIRegister_VectorLists(CL: TPSPascalCompiler);
13338: begin
13339:   TBaseListOption', '( bloExternalMemory, bloSetCountResetsMemory)
13340:   TBaseListOptions', 'set of TBaseListOption
13341:   SIRegister_TBaseList(CL);
13342:   SIRegister_TBaseVectorList(CL);
13343:   SIRegister_TAffineVectorList(CL);
13344:   SIRegister_TVectorList(CL);
13345:   SIRegister_TTexPointList(CL);
13346:   SIRegister_TXIntegerList(CL);
13347:   //PSingleArrayList', 'TSingleArrayList // will not work
13348:   SIRegister_TSingleList(CL);
13349:   SIRegister_TByteList(CL);
13350:   SIRegister_TQuaternionList(CL);
13351:   Procedure QuickSortLists( startIndex, endIndex : Integer; refList : TSingleList; objList : TList);
13352:   Procedure QuickSortLists1( startIndex, endIndex : Integer; refList : TSingleList; objList : TBaseList);
13353:   Procedure FastQuickSortLists(startIndex,
endIndex:Integer;refList:TSingleList;objList:TPersistentObjectList);
13354: end;
13355:
13356: procedure SIRegister_MeshUtils(CL: TPSPascalCompiler);

```

```

13357: begin
13358: Procedure ConvertStripToList( const strip : TAffineVectorList; list : TAffineVectorList);
13359: Procedure ConvertStripToList1( const strip : TIntegerList; list : TIntegerList);
13360: Procedure ConvertStripToList2(const strip:TAffineVectorList;const
indices:TIntegerList;list:TAffineVectorList);
13361: Procedure ConvertIndexedListToList(const data:TAffineVectlist;const
indices:TIntegerList;list:TAffineVectorList);
13362: Function BuildVectorCountOptimizedIndices(const vertices:TAffineVectorList; const
normals:TAffineVectorList; const texCoords : TAffineVectorList) : TIntegerList
13363: Procedure RemapReferences( reference : TAffineVectorList; const indices : TIntegerList);
13364: Procedure RemapReferences1( reference : TIntegerList; const indices : TIntegerList);
13365: Procedure RemapAndCleanupReferences( reference : TAffineVectorList; indices : TIntegerList)
13366: Function RemapIndicesToIndicesMap( remapIndices : TIntegerList) : TIntegerList
13367: Procedure RemapTrianglesIndices( indices, indicesMap : TIntegerList)
13368: Procedure RemapIndices( indices, indicesMap : TIntegerList)
13369: Procedure UnifyTrianglesWinding( indices : TIntegerList)
13370: Procedure InvertTrianglesWinding( indices : TIntegerList)
13371: Function BuildNormals( reference : TAffineVectorList; indices : TIntegerList) : TAffineVectorList
13372: Function BuildNonOrientedEdgesList(triangleIndices:TIntegerList; triangleEdges : TIntegerList;
edgesTriangles : TIntegerList) : TIntegerList
13373: Procedure WeldVertices( vertices : TAffineVectorList; indicesMap : TIntegerList; weldRadius : Single)
13374: Function StripifyMesh(indices:TIntegerList;maxVertexIndex:Integer;agglomerateLoneTriangles:Boolean);
TPersistentObjectList;
13375: Procedure IncreaseCoherency( indices : TIntegerList; cacheSize : Integer)
13376: Procedure SubdivideTriangles( smoothFactor : Single; vertices : TAffineVectorList; triangleIndices :
TIntegerList; normals : TAffineVectorList; onSubdivideEdge : TSubdivideEdgeEvent)
13377: end;
13378:
13379: procedure SIRegister_JclSysUtils(CL: TPSPascalCompiler);
13380: begin
13381: Procedure GetAndFillMem( var P : TObject; const Size : Integer; const Value : Byte)
13382: Procedure FreeMemAndNil( var P : TObject)
13383: Function PCharOrNil( const S : string) : PChar
13384: SIRegister_TJclReferenceMemoryStream(CL);
13385: FindClass('TObject','EJclVMError
13386: {Function GetVirtualMethodCount( AClass : TClass) : Integer
13387: Function GetVirtualMethod( AClass : TClass; const Index : Integer) : Pointer
13388: Procedure SetVirtualMethod( AClass : TClass; const Index : Integer; const Method:Pointer)
13389: PDynamicIndexList', '^TDynamicIndexList // will not work
13390: PDynamicAddressList', '^TDynamicAddressList // will not work
13391: Function GetDynamicMethodCount( AClass : TClass) : Integer
13392: Function GetDynamicIndexList( AClass : TClass) : PDynamicIndexList
13393: Function GetDynamicAddressList( AClass : TClass) : PDynamicAddressList
13394: Function HasDynamicMethod( AClass : TClass; Index : Integer) : Boolean
13395: Function GetDynamicMethod( AClass : TClass; Index : Integer) : Pointer
13396: Function GetInitTable( AClass : TClass) : PTypeInfo
13397: PFieldEntry', '^TFieldEntry // will not work}
13398: TFieldEntry', 'record OffSet : Integer; IDX : Word; Name : Short'
13399: + 'String; end
13400: Function JIsClass( Address : Pointer) : Boolean
13401: Function JIsObject( Address : Pointer) : Boolean
13402: Function GetImplementorOfInterface( const I : IInterface) : TObject
13403: TDigitCount', 'Integer
13404: SIRegister_TJclNumericFormat(CL);
13405: Function JIntToStrZeroPad( Value, Count : Integer) : AnsiString
13406: TTextHandler', 'Procedure ( const Text : string)
13407: // 'ABORT_EXIT_CODE','LongInt').SetInt( ERROR_CANCELLED 1223);
13408: Function JExecute(const
CommandLine:string;OutputLineCallback:TTextHandler;RawOutput:Bool;AbortPtr:PBool):Card;
13409: Function JExecutел(const CommandLine:string;var Output:string; RawOutput:Bool; AbortPtr:PBool):Cardinal;
13410: Function ReadKey : Char //to and from the DOS console !
13411: TModuleHandle', 'HINST
13412: //TModuleHandle', 'Pointer
13413: 'INVALID_MODULEHANDLE_VALUE','LongInt').SetInt( TModuleHandle ( 0 ));
13414: Function LoadModule( var Module : TModuleHandle; FileName : string) : Boolean
13415: Function LoadModuleEx( var Module : TModuleHandle; FileName : string; Flags : Cardinal) : Boolean
13416: Procedure UnloadModule( var Module : TModuleHandle)
13417: Function GetModuleSymbol( Module : TModuleHandle; SymbolName : string) : Pointer
13418: Function GetModuleSymbolEx( Module : TModuleHandle; SymbolName : string; var Accu : Boolean) : Pointer
13419: Function ReadModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size: Cardinal):Boolean;
13420: Function WriteModuleData(Module:TModuleHandle;SymbolName:string;var Buffer,Size:Cardinal):Boolean;
13421: FindClass('TObject','EJclConversionError
13422: Function JStrToBoolean( const S : string) : Boolean
13423: Function JBooleanToStr( B : Boolean) : string
13424: Function JIntToBool( I : Integer) : Boolean
13425: Function JBoolToInt( B : Boolean) : Integer
13426: 'ListSeparator','String').SetString( ' ;
13427: 'ListSeparator1','String').SetString( ' :
13428: Procedure ListAddItems( var List : string; const Separator, Items : string)
13429: Procedure ListIncludeItems( var List : string; const Separator, Items : string)
13430: Procedure ListRemoveItems( var List : string; const Separator, Items : string)
13431: Procedure ListDelItem( var List : string; const Separator : string; const Index : Integer)
13432: Function ListItemCount( const List, Separator : string) : Integer
13433: Function ListGetItem( const List, Separator : string; const Index : Integer) : string
13434: Procedure ListSetItem(var List:string;const Separator:string;const Index:Integer;const Value:string)
13435: Function ListItemIndex( const List, Separator, Item : string) : Integer
13436: Function SystemToObjectInstance : LongWord
13437: Function IsCompiledWithPackages : Boolean
13438: Function JclGUIDToString( const GUID : TGUID) : string

```



```

13439:   Function JJclStringToGUID( const S : string ) : TGUID
13440:   SIRegister_TJclIntfCriticalSection(CL);
13441:   SIRegister_TJclSimpleLog(CL);
13442:   Procedure InitSimpleLog( const ALogFileName : string)
13443:   end;
13444:
13445: procedure SIRegister_JclBorlandTools(CL: TPSPascalCompiler);
13446: begin
13447:   FindClass('TOBJECT'),'EJclBorRAException
13448:   TJclBorRADToolKind','( brDelphi, brCppBuilder, brBorlandDevStudio )
13449:   TJclBorRADToolEdition','( deOPEN, dePRO, deSVR )
13450:   TJclBorRADToolEdition','( deSTD, dePRO, deCSS, deARC )
13451:   TJclBorRADToolPath','string
13452:   'SupportedDelphiVersions','LongInt').SetInt( 5 or 6 or 7 or 8 or 9 or 10 or 11);
13453:   'SupportedBCBVersions','LongInt').SetInt( 5 or 6 or 10 or 11);
13454:   'SupportedBDSVersions','LongInt').SetInt( 1 or 2 or 3 or 4 or 5);
13455:   BorRADToolRepositoryPagesSection','String').SetString( 'Repository Pages
13456:   BorRADToolRepositoryDialogsPage','String').SetString( 'Dialogs
13457:   BorRADToolRepositoryFormsPage','String').SetString( 'Forms
13458:   BorRADToolRepositoryProjectsPage','String').SetString( 'Projects
13459:   BorRADToolRepositoryDataModulesPage','String').SetString( 'Data Modules
13460:   BorRADToolRepositoryObjectType','String').SetString( 'Type
13461:   BorRADToolRepositoryFormTemplate','String').SetString( 'FormTemplate
13462:   BorRADToolRepositoryProjectTemplate','String').SetString( 'ProjectTemplate
13463:   BorRADToolRepositoryObjectName','String').SetString( 'Name
13464:   BorRADToolRepositoryObjectPage','String').SetString( 'Page
13465:   BorRADToolRepositoryObjectIcon','String').SetString( 'Icon
13466:   BorRADToolRepositoryObjectDescr','String').SetString( 'Description
13467:   BorRADToolRepositoryObjectAuthor','String').SetString( 'Author
13468:   BorRADToolRepositoryObjectAncestor','String').SetString( 'Ancestor
13469:   BorRADToolRepositoryObjectDesigner','String').SetString( 'Designer
13470:   BorRADToolRepositoryDesignerDfm','String').SetString( 'dfm
13471:   BorRADToolRepositoryDesignerXfm','String').SetString( 'xfm
13472:   BorRADToolRepositoryObjectNewForm','String').SetString( 'DefaultNewForm
13473:   BorRADToolRepositoryObjectMainForm','String').SetString( 'DefaultMainForm
13474:   SourceExtensionDelphiPackage','String').SetString( '.dpk
13475:   SourceExtensionBCBPackage','String').SetString( '.bpc
13476:   SourceExtensionDelphiProject','String').SetString( '.dpr
13477:   SourceExtensionBCBProject','String').SetString( '.bpr
13478:   SourceExtensionBDSProject','String').SetString( '.bdsproj
13479:   SourceExtensionDProject','String').SetString( '.dproj
13480:   BinaryExtensionPackage','String').SetString( '.bpl
13481:   BinaryExtensionLibrary','String').SetString( '.dll
13482:   BinaryExtensionExecutable','String').SetString( '.exe
13483:   CompilerExtensionDCP','String').SetString( '.dcp
13484:   CompilerExtensionBPI','String').SetString( '.bpi
13485:   CompilerExtensionLIB','String').SetString( '.lib
13486:   CompilerExtensionTDS','String').SetString( '.tds
13487:   CompilerExtensionMAP','String').SetString( '.map
13488:   CompilerExtensionDRC','String').SetString( '.drc
13489:   CompilerExtensionDEF','String').SetString( '.def
13490:   SourceExtensionCPP','String').SetString( '.cpp
13491:   SourceExtensionH','String').SetString( '.h
13492:   SourceExtensionPAS','String').SetString( '.pas
13493:   SourceExtensionDFM','String').SetString( '.dfm
13494:   SourceExtensionXFM','String').SetString( '.xfm
13495:   SourceDescriptionPAS','String').SetString( 'Pascal source file
13496:   SourceDescriptionCPP','String').SetString( 'C++ source file
13497:   DesignerVCL','String').SetString( 'VCL
13498:   DesignerCLX','String').SetString( 'CLX
13499:   ProjectTypePackage','String').SetString( 'package
13500:   ProjectTypeLibrary','String').SetString( 'library
13501:   ProjectTypeProgram','String').SetString( 'program
13502:   Personality32Bit','String').SetString( '32 bit
13503:   Personality64Bit','String').SetString( '64 bit
13504:   PersonalityDelphi','String').SetString( 'Delphi
13505:   PersonalityDelphiDotNet','String').SetString( 'Delphi.net
13506:   PersonalityBCB','String').SetString( 'C++Builder
13507:   PersonalityCSB','String').SetString( 'C#Builder
13508:   PersonalityVB','String').SetString( 'Visual Basic
13509:   PersonalityDesign','String').SetString( 'Design
13510:   PersonalityUnknown','String').SetString( 'Unknown personality
13511:   PersonalityBDS','String').SetString( 'Borland Developer Studio
13512:   DOFDirectoriesSection','String').SetString( 'Directories
13513:   DOFUnitOutputDirKey','String').SetString( 'UnitOutputDir
13514:   DOFSearchPathName','String').SetString( 'SearchPath
13515:   DOFConditionals','String').SetString( 'Conditionals
13516:   DOFLinkerSection','String').SetString( 'Linker
13517:   DOFPackagesKey','String').SetString( 'Packages
13518:   DOFCompilerSection','String').SetString( 'Compiler
13519:   DOFPackageNoLinkKey','String').SetString( 'PackageNoLink
13520:   DOFAdditionalSection','String').SetString( 'Additional
13521:   DOFOptionsKey','String').SetString( 'Options
13522:   TJclBorPersonality','( bpDelphi32, bpDelphi64, bpBCBuilder32, b'
13523:   + 'pBCBuilder64, bpDelphiNet32, bpDelphiNet64, bpCSBuilder32, bpCSBuilder64, '
13524:   + 'bpVisualBasic32, bpVisualBasic64, bpDesign, bpUnknown )
13525:   TJclBorPersonalities','set of TJclBorPersonality
13526:   TJclBorDesigner','( bdVCL, bdCLX )
13527:   TJclBorDesigners','set of TJclBorDesigner

```

```

13528: TJclBorPlatform', '( bp32bit, bp64bit )
13529: FindClass('TOBJECT'), 'TJclBorRADToolInstallation
13530: SIRegister_TJclBorRADToolInstallationObject(CL);
13531: SIRegister_TJclBorlandOpenHelp(CL);
13532: TJclHelp2Object', '( hoRegisterSession, hoRegister, hoPlugin )
13533: TJclHelp2Objects', 'set of TJclHelp2Object
13534: SIRegister_TJclHelp2Manager(CL);
13535: SIRegister_TJclBorRADToolIdeTool(CL);
13536: SIRegister_TJclBorRADToolIdePackages(CL);
13537: SIRegister_IJclCommandLineTool(CL);
13538: FindClass('TOBJECT'), 'EJclCommandLineToolError
13539: SIRegister_TJclCommandLineTool(CL);
13540: SIRegister_TJclBorlandCommandLineTool(CL);
13541: SIRegister_TJclBCC32(CL);
13542: SIRegister_TJclDCC32(CL);
13543: TJclDCC', 'TJclDCC32
13544: SIRegister_TJclBpr2Mak(CL);
13545: SIRegister_TJclBorlandMake(CL);
13546: SIRegister_TJclBorRADToolPalette(CL);
13547: SIRegister_TJclBorRADToolRepository(CL);
13548: TCommandLineTool', '( clAsm, clBcc32, clDcc32, clDccIL, clMake, clProj2Mak )
13549: TCommandLineTools', 'set of TCommandLineTool
13550: //TJclBorRADToolInstallationClass', 'class of TJclBorRADToolInstallation
13551: SIRegister_TJclBorRADToolInstallation(CL);
13552: SIRegister_TJclBCBInstallation(CL);
13553: SIRegister_TJclDelphiInstallation(CL);
13554: SIRegister_TJclDCCIL(CL);
13555: SIRegister_TJclBDSInstallation(CL);
13556: TTraverseMethod', 'Function ( Installation : TJclBorRADToolInstallation ) : Boolean
13557: SIRegister_TJclBorRADToolInstallations(CL);
13558: Function BPLFileName( const BPLPath, PackageFileName : string ) : string
13559: Function BinaryFileName( const OutputPath, ProjectFileName : string ) : string
13560: Function IsDelphiPackage( const FileName : string ) : Boolean
13561: Function IsDelphiProject( const FileName : string ) : Boolean
13562: Function IsBCBPackage( const FileName : string ) : Boolean
13563: Function IsBCBProject( const FileName : string ) : Boolean
13564: Procedure GetDPKFileInfo(const DPRFileName:string;out BinaryExtensio:string;const LibSuffix:PString);
13565: Procedure GetBPRFileInfo(const BPRFileName:string;out BinaryFileName:string;const Descript:PString);
13566: Procedure GetDPKFileInfo(const DPKFileName:string;out RunOnly:Bool;const LibSuffix:PString;const
Descript:PString;
13567: Procedure GetBPKFileInfo(const BPKFileName:string;out RunOnly:Bool;const BinaryFNme:PString;const
Descript:PString);
13568: function SamePath(const Path1, Path2: string): Boolean;
13569: end;
13570:
13571: procedure SIRegister_JclFileUtils_max(CL: TPSPascalCompiler);
13572: begin
13573: 'ERROR_NO_MORE_FILES', 'LongInt').SetInt( 18);
13574: //Function stat64( FileName: PChar;var StatBuffer : TStatBuf64 ) : Integer
13575: //Function fstat64( FileDes: Integer;var StatBuffer : TStatBuf64 ) : Integer
13576: //Function lstat64( FileName: PChar;var StatBuffer : TStatBuf64 ) : Integer
13577: 'LPathSeparator', 'String').SetString( '/'
13578: 'LDirDelimiter', 'String').SetString( '/'
13579: 'LDirSeparator', 'String').SetString( ':'
13580: 'JXPathDevicePrefix', 'String').SetString( '\\.\
13581: 'JXPathSeparator', 'String').SetString( '\
13582: 'JXDirDelimiter', 'String').SetString( '\
13583: 'JXDirSeparator', 'String').SetString( ';'
13584: 'JXPathUncPrefix', 'String').SetString( '\\
13585: 'faNormalFile', 'LongWord').SetUInt( $00000080);
13586: //faUnixSpecific', '').SetString( faSymLink);
13587: JXTCompactPath', '( cpCenter, cpEnd )
13588: _WIN32_FILE_ATTRIBUTE_DATA', 'record dwFileAttributes : DWORD; f
13589: + 'tCreationTime : TFileTime; ftLastAccessTime : TFileTime; ftLastWriteTime :
13590: + ' TFileTime; nFileSizeHigh : DWORD; nFileSizeLow : DWORD; end
13591: TWin32FileAttributeData', '_WIN32_FILE_ATTRIBUTE_DATA
13592: WIN32_FILE_ATTRIBUTE_DATA', '_WIN32_FILE_ATTRIBUTE_DATA
13593:
13594: Function jxPathAddSeparator( const Path : string ) : string
13595: Function jxPathAddExtension( const Path, Extension : string ) : string
13596: Function jxPathAppend( const Path, Append : string ) : string
13597: Function jxPathBuildRoot( const Drive : Byte ) : string
13598: Function jxPathCanonicalize( const Path : string ) : string
13599: Function jxPathCommonPrefix( const Path1, Path2 : string ) : Integer
13600: Function jxPathCompactPath(const DC:HDC;const Path:string;const Width:Integer;CmpFmt:TCompactPath):string
13601: Procedure jxPathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
13602: Function jxPathExtractFileDirFixed( const S : string ) : string
13603: Function jxPathExtractFileNameNoExt( const Path : string ) : string
13604: Function jxPathExtractPathDepth( const Path : string; Depth : Integer ) : string
13605: Function jxPathGetDepth( const Path : string ) : Integer
13606: Function jxPathGetLongName( const Path : string ) : string
13607: Function jxPathGetShortName( const Path : string ) : string
13608: Function jxPathGetLongName( const Path : string ) : string
13609: Function jxPathGetShortName( const Path : string ) : string
13610: Function jxPathGetRelativePath( Origin, Destination : string ) : string
13611: Function jxPathGetTempPath : string
13612: Function jxPathIsAbsolute( const Path : string ) : Boolean
13613: Function jxPathIsChild( const Path, Base : string ) : Boolean
13614: Function jxPathIsDiskDevice( const Path : string ) : Boolean

```

```

13615: Function jxPathIsUNC( const Path : string) : Boolean
13616: Function jxPathRemoveSeparator( const Path : string) : string
13617: Function jxPathRemoveExtension( const Path : string) : string
13618: Function jxPathGetPhysicalPath( const LocalizedPath : string) : string
13619: Function jxPathGetLocalizedPath( const PhysicalPath : string) : string
13620:   JxTFileListOption', '( flFullNames, flRecursive, flMaskedSubfolders)
13621:   JxTFileListOptions', 'set of TFileListOption
13622:   JxTJclAttributeMatch', '( amAny, amExact, amSubSetOf, amSuperSetOf, amCustom )
13623:   TFileHandler', 'Procedure ( const FileName : string)
13624:   TFileHandlerEx', 'Procedure ( const Directory : string; const FileInfo : TSearchRec)
13625: Function BuildFileList( const Path : string; const Attr : Integer; const List : TStrings) : Boolean
13626: //Function AdvBuildFileList( const Path : string; const Attr : Integer; const Files : TStrings; const
AttributeMatch:TJclAttributeMatch;const Optis:TFileListOptions;const SubfoldersMask:string;const
FileMatchFunc:TFileMatchFunc):Bool;
13627: Function jxVerifyFileAttributeMask( var RejectedAttributes, RequiredAttributes : Int) : Boolean
13628: Function jxIsFileAttributeMatch(FileAttributes,RejectedAttributes:Int):Boolean;
13629: Function jxFileAttributesStr( const FileInfo : TSearchRec) : string
13630: Function jxIsFileNameMatch(FileName:string;const Mask:string;const CaseSensitive:Boolean):Boolean;
13631: Procedure jxEnumFiles(const Path:string; HandleFile:TFileHandlerEx;
RejectedAttributes:Integer;RequiredAttributes : Integer; Abort : TObject)
13632: Procedure jxEnumDirectories(const Root:string;const HandleDirectory:TFileHandler;const
IncludeHiddenDirects:Boolean; const SubDirectoriesMask : string; Abort : TObject; ResolveSymLinks :
Boolean)
13633: Procedure jxCreateEmptyFile( const FileName : string)
13634: Function jxCloseVolume( var Volume : THandle) : Boolean
13635: Function jxDeleteDirectory( const DirectoryName : string; MoveToRecycleBin : Boolean) : Boolean
13636: Function jxCopyDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13637: Function jxMoveDirectory( ExistingDirectoryName, NewDirectoryName : string) : Boolean
13638: Function jxDelTree( const Path : string) : Boolean
13639: //Function DelTreeEx(const Path:string;AbortOnFailure:Boolean; Progress:TDelTreeProgress):Boolean
13640: Function jxDiskInDrive( Drive : Char) : Boolean
13641: Function jxDirectoryExists( const Name : string; ResolveSymLinks : Boolean) : Boolean
13642: Function jxFileCreateTemp( var Prefix : string) : THandle
13643: Function jxFileBackup( const FileName : string; Move : Boolean) : Boolean
13644: Function jxFileCopy( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13645: Function jxFileDelete( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13646: Function jxFileExists( const FileName : string) : Boolean
13647: Function jxFileMove( const ExistingFileName, NewFileName : string; ReplaceExisting : Boolean) : Boolean
13648: Function jxFileRestore( const FileName : string) : Boolean
13649: Function jxGetBackupFileName( const FileName : string) : string
13650: Function jxIsBackupFileName( const FileName : string) : Boolean
13651: Function jxFileGetDisplayName( const FileName : string) : string
13652: Function jxFileGetGroupName( const FileName : string; ResolveSymLinks : Boolean) : string
13653: Function jxFileGetOwnerName( const FileName : string; ResolveSymLinks : Boolean) : string
13654: Function jxFileGetSize( const FileName : string) : Int64
13655: Function jxFileGetTempName( const Prefix : string) : string
13656: Function jxFileGetType( const FileName : string) : string
13657: Function jxFindUnusedFileName(FileName:string; const FileExt : string; NumberPrefix : string) : string
13658: Function jxForceDirectories( Name : string) : Boolean
13659: Function jxGetDirectorySize( const Path : string) : Int64
13660: Function jxGetDriveTypeStr( const Drive : Char) : string
13661: Function jxGetFileAgeCoherence( const FileName : string) : Boolean
13662: Procedure jxGetFileAttributeList( const Items : TStrings; const Attr : Integer)
13663: Procedure jxGetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
13664: Function jxGetFileInformation( const FileName : string; out FileInfo : TSearchRec) : Boolean;
13665: Function jxGetFileInformation1( const FileName : string) : TSearchRec;
13666: //Function GetFileStatus(const FileName:string;out StatBuf:TStatBuf64;const
ResolveSymLinks:Boolean):Integer
13667: Function jxGetFileLastWrite( const FName : string) : TFileTime;
13668: Function jxGetFileLastWrite1( const FName : string; out LocalTime : TDateTime) : Boolean;
13669: Function jxGetFileLastAccess( const FName : string) : TFileTime;
13670: Function jxGetFileLastAccess1( const FName : string; out LocalTime : TDateTime) : Boolean;
13671: Function jxGetFileCreation( const FName : string) : TFileTime;
13672: Function jxGetFileCreation1( const FName : string; out LocalTime : TDateTime) : Boolean;
13673: Function jxGetFileLastWrite( const FName : string; out TimeStamp:Integer;ResolveSymLinks : Bool):Bool;
13674: Function jxGetFileLastWrite1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool): Bool;
13675: Function jxGetFileLastWrite2( const FName : string; ResolveSymLinks : Boolean) : Integer;
13676: Function jxGetFileLastAccess(const FName:string; out TimeStamp:Integer;ResolveSymLinks: Bool): Bool;
13677: Function jxGetFileLastAccess1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13678: Function jxGetFileLastAccess2(const FName:string; ResolveSymLinks:Boolean): Integer;
13679: Function jxGetFileLastAttrChange(const FName:string;out TimeStamp:Integer;ResolveSymLinks:Bool): Bool;
13680: Function jxGetFileLastAttrChange1(const FName:string; out LocalTime:TDateTime;ResolveSymLinks:Bool):Bool;
13681: Function jxGetFileLastAttrChange2( const FName : string; ResolveSymLinks:Boolean): Integer;
13682: Function jxGetModulePath( const Module : HMODULE) : string
13683: Function jxGetSizeOfFile( const FileName : string) : Int64;
13684: Function jxGetSizeOfFile1( const FileInfo : TSearchRec) : Int64;
13685: Function jxGetSizeOfFile2( Handle : THandle) : Int64;
13686: Function jxGetStandardFileInfo( const FileName : string) : TWin32FileAttributeData
13687: Function jxIsDirectory( const FileName : string; ResolveSymLinks : Boolean) : Boolean
13688: Function jxIsRootDirectory( const CanonicFileName : string) : Boolean
13689: Function jxLockVolume( const Volume : string; var Handle : THandle) : Boolean
13690: Function jxOpenVolume( const Drive : Char) : THandle
13691: Function jxSetDirLastWrite( const DirName : string; const DateTime : TDateTime) : Boolean
13692: Function jxSetDirLastAccess( const DirName : string; const DateTime : TDateTime) : Boolean
13693: Function jxSetDirCreation( const DirName : string; const DateTime : TDateTime) : Boolean
13694: Function jxSetFileLastWrite( const FileName : string; const DateTime : TDateTime) : Boolean
13695: Function jxSetFileLastAccess( const FileName : string; const DateTime : TDateTime) : Boolean
13696: Function jxSetFileCreation( const FileName : string; const DateTime : TDateTime) : Boolean
13697: Procedure jxShredFile( const FileName : string; Times : Integer)

```

```

13698: Function jxUnlockVolume( var Handle : THandle) : Boolean
13699: Function jxCreatSymbolicLink( const Name, Target : string) : Boolean
13700: Function jxSymbolicLinkTarget( const Name : string) : string
13701: TAttributeInterest', '( aiIgnored, aiRejected, aiRequired )
13702: SIRegister_TJclCustomFileAttrMask(CL);
13703: SIRegister_TJclFileAttributeMask(CL);
13704: TFileSearchOption', '( fsIncludeSubDirectories, fsIncludeHiddenS'
13705: + 'ubDirectories, fsLastChangeAfter, fsLastChangeBefore, fsMaxSize, fsMinSize)
13706: TFileSearchOptions', 'set of TFileSearchOption
13707: TFileSearchTaskID', 'Integer
13708: TFileSearchTerminationEvent', 'Procedure ( const ID : TFileSearch'
13709: + 'hTaskID; const Aborted : Boolean)
13710: TFileEnumeratorSyncMode', '( smPerFile, smPerDirectory )
13711: SIRegister_IJclFileEnumerator(CL);
13712: SIRegister_TJclFileEnumerator(CL);
13713: Function JxFileSearch : IJclFileEnumerator
13714: JxTFileFlag', '( ffDebug, ffInfoInferred, ffPatched, ffPreRelease, ffPrivateBuild, ffSpecialBuild )
13715: JxTFileFlags', 'set of TFileFlag
13716: FindClass('TOBJECT'), 'EJclFileVersionInfoError
13717: SIRegister_TJclFileVersionInfo(CL);
13718: Function jxOSIdentToString( const OSIdent : DWORD) : string
13719: Function jxOSFileTypeToString( const OSFileType : DWORD; const OSFileSubType : DWORD) : string
13720: Function jxVersionResourceAvailable( const FileName : string) : Boolean
13721: TFileVersionFormat', '( vfMajorMinor, vfFull )
13722: Function jxFormatVersionString( const HiV, LoV : Word) : string;
13723: Function jxFormatVersionString1( const Major, Minor, Build, Revision : Word) : string;
13724: //Function FormatVersionString2( const FixedInfo : TVSFixedFileInfo; VersionFormat: TFileVersionFormat):str;
13725: //Procedure VersionExtractFileInfo(const FixedInfo: TVSFixedFileInfo; var Major, Minor, Build, Revision: Word);
13726: //Procedure VersionExtractProductInfo(const FixedInfo: TVSFixedFileInfo; var Major, Minor, Build,
Revision: Word);
13727: //Function VersionFixedFileInfo( const FileName : string; var FixedInfo : TVSFixedFileInfo) : Boolean
13728: Function jxVersionFixedFileInfoString( const FileName : string; VersionFormat : TFileVersionFormat; const
NotAvailableText : string) : string
13729: SIRegister_TJclTempFileStream(CL);
13730: FindClass('TOBJECT'), 'TJclCustomFileMapping
13731: SIRegister_TJclFileMappingView(CL);
13732: TJclFileMappingRoundOffset', '( rvDown, rvUp )
13733: SIRegister_TJclCustomFileMapping(CL);
13734: SIRegister_TJclFileMapping(CL);
13735: SIRegister_TJclSwapFileMapping(CL);
13736: SIRegister_TJclFileMappingStream(CL);
13737: TJclMappedTextReaderIndex', '( tiNoIndex, tiFull )
13738: //PPCharArray', '^TPCharArray // will not work
13739: SIRegister_TJclMappedTextReader(CL);
13740: SIRegister_TJclFileMaskComparator(CL);
13741: FindClass('TOBJECT'), 'EJclPathError
13742: FindClass('TOBJECT'), 'EJclFileUtilsError
13743: FindClass('TOBJECT'), 'EJclTempFileStreamError
13744: FindClass('TOBJECT'), 'EJclTempFileStreamError
13745: FindClass('TOBJECT'), 'EJclFileMappingError
13746: FindClass('TOBJECT'), 'EJclFileMappingViewError
13747: Function jxPathGetLongName2( const Path : string) : string
13748: Function jxWin32DeleteFile( const FileName : string; MoveToRecycleBin : Boolean) : Boolean
13749: Function jxWin32MoveFileReplaceExisting( const SrcFileName, DstFileName : string) : Boolean
13750: Function jxWin32BackupFile( const FileName : string; Move : Boolean) : Boolean
13751: Function jxWin32RestoreFile( const FileName : string) : Boolean
13752: Function jxSamePath( const Path1, Path2 : string) : Boolean
13753: Procedure jxPathListAddItems( var List : string; const Items : string)
13754: Procedure jxPathListIncludeItems( var List : string; const Items : string)
13755: Procedure jxPathListDelItems( var List : string; const Items : string)
13756: Procedure jxPathListDelItem( var List : string; const Index : Integer)
13757: Function jxPathListItemCount( const List : string) : Integer
13758: Function jxPathListGetItem( const List : string; const Index : Integer) : string
13759: Procedure jxPathListSetItem( var List : string; const Index : Integer; const Value : string)
13760: Function jxPathListItemIndex( const List, Item : string) : Integer
13761: Function jxParamName(Idx: Int; const Separator: string; const AllowedPrefixChars: string; TrimName: Bool): string;
13762: Function jxParamValue(Index : Integer; const Separator : string; TrimValue : Boolean) : string;
13763: Function jxParamValue1(const SearchName: string; const Separator : string; CaseSensitive : Boolean; const
AllowedPrefixCharacters : string; TrimValue : Boolean) : string;
13764: Function jxParamPos( const SearchName : string; const Separator : string; CaseSensitive : Boolean; const
AllowedPrefixCharacters : string) : Integer
13765: end;
13766:
13767: procedure SIRegister_FileUtil(CL: TPSPascalCompiler);
13768: begin
13769:   'UTF8FileHeader', 'String').SetString( #Sef#Sbb#Sbf);
13770: Function lCompareFileNames( const Filename1, Filename2 : string) : integer
13771: Function lCompareFileNamesIgnoreCase( const Filename1, Filename2 : string) : integer
13772: Function lCompareFileNames( const Filename1, Filename2 : string; ResolveLinks : boolean) : integer
13773: Function lCompareFileNames(Filename1: PChar; Len1: int; Filename2: PChar; Len2: int; ResolveLinks: boolean): int;
13774: Function lFilenameIsAbsolute( const TheFilename : string) : boolean
13775: Function lFilenameIsWinAbsolute( const TheFilename : string) : boolean
13776: Function lFilenameIsUnixAbsolute( const TheFilename : string) : boolean
13777: Procedure lCheckIfFileIsExecutable( const AFilename : string)
13778: Procedure lCheckIfFileIsSymlink( const AFilename : string)
13779: Function lFileIsReadable( const AFilename : string) : boolean
13780: Function lFileIsWritable( const AFilename : string) : boolean
13781: Function lFileIsText( const AFilename : string) : boolean
13782: Function lFileIsText( const AFilename : string; out FileReadable : boolean) : boolean

```



```

13783: Function lFileIsExecutable( const AFilename : string ) : boolean
13784: Function lFileIsSymlink( const AFilename : string ) : boolean
13785: Function lFileIsHardLink( const AFilename : string ) : boolean
13786: Function lFileSize( const Filename : string ) : int64;
13787: Function lGetFileDescription( const AFilename : string ) : string
13788: Function lReadAllLinks( const Filename : string; ExceptionOnError : boolean ) : string
13789: Function lTryReadAllLinks( const Filename : string ) : string
13790: Function lDirPathExists( const FileName : String ) : Boolean
13791: Function lForceDirectory( DirectoryName : string ) : boolean
13792: Function lDeleteDirectory( const DirectoryName : string; OnlyChildren : boolean ) : boolean
13793: Function lProgramDirectory : string
13794: Function lDirectoryIsWritable( const DirectoryName : string ) : boolean
13795: Function lExtractFileNameOnly( const AFilename : string ) : string
13796: Function lExtractFileNameWithoutExt( const AFilename : string ) : string
13797: Function lCompareFileExt( const Filename, Ext : string; CaseSensitive : boolean ) : integer;
13798: Function lCompareFileExt( const Filename, Ext : string ) : integer;
13799: Function lFilenameIsPascalUnit( const Filename : string ) : boolean
13800: Function lAppendPathDelim( const Path : string ) : string
13801: Function lChompPathDelim( const Path : string ) : string
13802: Function lTrimFilename( const AFilename : string ) : string
13803: Function lCleanAndExpandFilename( const Filename : string ) : string
13804: Function lCleanAndExpandDirectory( const Filename : string ) : string
13805: Function lCreateAbsolutePath( const SearchPath, BaseDirectory : string ) : string
13806: Function lCreateRelativePath( const Filename, BaseDirectory : string; UsePointDirectory : boolean;
AlwaysRequireSharedBaseFolder : Boolean ) : string
13807: Function lCreateAbsolutePath( const Filename, BaseDirectory : string ) : string
13808: Function lFileIsInPath( const Filename, Path : string ) : boolean
13809: Function lFileIsInDirectory( const Filename, Directory : string ) : boolean
13810: TSearchFileInPathFlag, '( sffDontSearchInBasePath, sffSearchLoUpCase )
13811: TSearchFileInPathFlags, 'set of TSearchFileInPathFlag
13812: 'AllDirectoryEntriesMask', 'String').SetString( '*'
13813: Function lGetAllFilesMask : string
13814: Function lGetExeExt : string
13815: Function lSearchFileInPath( const Filename, BasePath, SearchPath, Delimiter : string; Flags :
TSearchFileInPathFlags ) : string
13816: Function lSearchAllFilesInPath( const Filename, BasePath, SearchPath, Delimiter:string;Flags :
TSearchFileInPathFlags ) : TStringList
13817: Function lFindDiskFilename( const Filename : string ) : string
13818: Function lFindDiskFileCaseInsensitive( const Filename : string ) : string
13819: Function lFindDefaultExecutablePath( const Executable : string; const BaseDir: string):string
13820: Function lGetDarwinSystemFilename( Filename : string ) : string
13821: SIRegister_TFileIterator(CL);
13822: TFileFoundEvent, 'Procedure ( FileIterator : TFileIterator)
13823: TDirectoryFoundEvent, 'Procedure ( FileIterator : TFileIterator)
13824: TDirectoryEnterEvent, 'Procedure ( FileIterator : TFileIterator)
13825: SIRegister_TFileSearcher(CL);
13826: Function lFindAllFiles(const SearchPath:String;SearchMsk:String;SearchSubDirs:Bool):TStringList
13827: Function lFindAllDirectories( const SearchPath : string; SearchSubDirs : Boolean ) : TStringList
13828: // TCopyFileFlag, '( cffOverwriteFile, cffCreateDestDirectory, cffPreserveTime )
13829: // TCopyFileFlags, 'set of TCopyFileFlag
13830: Function lCopyFile( const SrcFilename, DestFilename : string; Flags : TCopyFileFlags ) : boolean
13831: Function lCopyFile( const SrcFilename, DestFilename : string; PreserveTime : boolean ) : boolean
13832: Function lCopyDirTree( const SourceDir, TargetDir : string; Flags : TCopyFileFlags ) : Boolean
13833: Function lReadFileToString( const Filename : string ) : string
13834: Function lGetTempFilename( const Directory, Prefix : string ) : string
13835: {Function NeedRTLAnsi : boolean
13836: Procedure SetNeedRTLAnsi( NewValue : boolean)
13837: Function UTF8ToSys( const s : string ) : string
13838: Function SysToUTF8( const s : string ) : string
13839: Function ConsoleToUTF8( const s : string ) : string
13840: Function UTF8ToConsole( const s : string ) : string}
13841: Function FileExistsUTF8( const Filename : string ) : boolean
13842: Function FileAgeUTF8( const FileName : string ) : Longint
13843: Function DirectoryExistsUTF8( const Directory : string ) : Boolean
13844: Function ExpandFileNameUTF8( const FileName : string ) : string
13845: Function ExpandUNCFileNameUTF8( const FileName : string ) : string
13846: Function ExtractShortPathNameUTF8( const FileName : String ) : String
13847: Function FindFirstUTF8( const Path : string; Attr : Longint; out Rslt : TSearchRec ) : Longint
13848: Function FindNextUTF8( var Rslt : TSearchRec ) : Longint
13849: Procedure FindCloseUTF8( var F : TSearchrec)
13850: Function FileSetDateUTF8( const FileName : String; Age : Longint ) : Longint
13851: Function FileGetAttrUTF8( const FileName : String ) : Longint
13852: Function FileSetAttrUTF8( const FileName : String; Attr : longint ) : Longint
13853: Function DeleteFileUTF8( const FileName : String ) : Boolean
13854: Function RenameFileUTF8( const OldName, NewName : String ) : Boolean
13855: Function FileSearchUTF8( const Name, DirList : String; ImplicitCurrentDir : Boolean ) : String
13856: Function FileIsReadOnlyUTF8( const FileName : String ) : Boolean
13857: Function GetCurrentDirUTF8 : String
13858: Function SetCurrentDirUTF8( const NewDir : String ) : Boolean
13859: Function CreateDirUTF8( const NewDir : String ) : Boolean
13860: Function RemoveDirUTF8( const Dir : String ) : Boolean
13861: Function ForceDirectoriesUTF8( const Dir : string ) : Boolean
13862: Function FileOpenUTF8( const FileName : string; Mode : Integer ) : THandle
13863: Function FileCreateUTF8( const FileName : string ) : THandle;
13864: Function FileCreateUTF8( const FileName : string; Rights : Cardinal ) : THandle;
13865: Function ParamStrUTF8( Param : Integer ) : string
13866: Function GetEnvironmentStringUTF8( Index : Integer ) : string
13867: Function GetEnvironmentVariableUTF8( const EnvVar : string ) : String
13868: Function GetAppConfigDirUTF8( Global : Boolean; Create : boolean ) : string

```

```

13869: Function GetAppConfigFileUTF8(Global:Boolean; SubDir:boolean; CreateDir : boolean) : string
13870: Function SysErrorMessageUTF8( ErrorCode : Integer) : String
13871: end;
13872:
13873: procedure SIRegister_Keyboard(CL: TPSPascalCompiler);
13874: begin
13875:     //VK_F23 = 134;
13876:     //{$EXTERNALSYM VK_F24}
13877:     //VK_F24 = 135;
13878:     TVirtualKeyCode, 'Integer
13879:     'VK_MOUSEWHEELUP', 'integer').SetInt(134);
13880:     'VK_MOUSEWHEELDOWN', 'integer').SetInt(135);
13881:     Function glIsKeyDown( c : Char) : Boolean;
13882:     Function glIsKeyDown( vk : TVirtualKeyCode) : Boolean;
13883:     Function glKeyPressed( minVkCode : TVirtualKeyCode) : TVirtualKeyCode
13884:     Function glVirtualKeyCodeToKeyName( vk : TVirtualKeyCode) : String
13885:     Function glKeyNameToVirtualKeyCode( const keyName : String) : TVirtualKeyCode
13886:     Function glCharToVirtualKeyCode( c : Char) : TVirtualKeyCode
13887:     Procedure glKeyboardNotifyWheelMoved( wheelDelta : Integer)
13888: end;
13889:
13890: procedure SIRegister_GLCrossPlatform(CL: TPSPascalCompiler);
13891: begin
13892:     TGLPoint, 'TPoint
13893:     //PGLPoint', '^TGLPoint // will not work
13894:     TGLRect, 'TRect
13895:     //PGLRect', '^TGLRect // will not work
13896:     TDelphiColor, 'TColor
13897:     TGLPicture, 'TPicture
13898:     TGLGraphic, 'TGraphic
13899:     TGLBitmap, 'TBitmap
13900:     //TGraphicClass', 'class of TGraphic
13901:     TGLTextLayout, '( tlTop, tlCenter, tlBottom )
13902:     TGLMouseButton, '( mbLeft, mbRight, mbMiddle )
13903:     TGLMouseEvent, 'Procedure ( Sender : TObject; Button : TGLMouse'
13904:         + 'Button; Shift : TShiftState; X, Y : Integer)
13905:     TGLMouseMoveEvent, 'TMouseMoveEvent
13906:     TGLKeyEvent, 'TKeyEvent
13907:     TGLKeyPressEvent, 'TKeyPressEvent
13908:     EGLOSError, 'EWin32Error
13909:     EGLOSError, 'EWin32Error
13910:     EGLOSError, 'EOSError
13911:     'glsAllFilter', 'string').SetString('All // sAllFilter
13912:     Function GLPoint( const x, y : Integer) : TGLPoint
13913:     Function GLRGB( const r, g, b : Byte) : TColor
13914:     Function GLColorToRGB( color : TColor) : TColor
13915:     Function GLGetRValue( rgb : DWORD) : Byte
13916:     Function GLGetGValue( rgb : DWORD) : Byte
13917:     Function GLGetBValue( rgb : DWORD) : Byte
13918:     Procedure GLInitWinColors
13919:     Function GLRect( const aLeft, aTop, aRight, aBottom : Integer) : TGLRect
13920:     Procedure GLInflateGLRect( var aRect : TGLRect; dx, dy : Integer)
13921:     Procedure GLIntersectGLRect( var aRect : TGLRect; const rect2 : TGLRect)
13922:     Procedure GLInformationDlg( const msg : String)
13923:     Function GLQuestionDlg( const msg : String) : Boolean
13924:     Function GLInputDialog( const aCaption, aPrompt, aDefault : String) : String
13925:     Function GLSavePictureDialog( var aFileName : String; const aTitle : String) : Boolean
13926:     Function GLOpenPictureDialog( var aFileName : String; const aTitle : String) : Boolean
13927:     Function GLApplicationTerminated : Boolean
13928:     Procedure GLRaiseLastOSError
13929:     Procedure GLFreeAndNil( var anObject: TObject)
13930:     Function GLGetDeviceLogicalPixelsX( device : Cardinal) : Integer
13931:     Function GLGetCurrentColorDepth : Integer
13932:     Function GLPixelFormatToColorBits( aPixelFormat : TPixelFormat) : Integer
13933:     Function GLBitmapScanLine( aBitmap : TGLBitmap; aRow : Integer) : Pointer
13934:     Procedure GLSleep( length : Cardinal)
13935:     Procedure GLQueryPerformanceCounter( var val : Int64)
13936:     Function GLQueryPerformanceFrequency( var val : Int64) : Boolean
13937:     Function GLStartPrecisionTimer : Int64
13938:     Function GLPrecisionTimerLap( const precisionTimer : Int64) : Double
13939:     Function GLStopPrecisionTimer( const precisionTimer : Int64) : Double
13940:     Function GLRDTSC : Int64
13941:     Procedure GLLoadBitmapFromInstance( ABitmap : TBitmap; AName : string)
13942:     Function GLOKMessageBox( const Text, Caption : string) : Integer
13943:     Procedure GLShowHTMLUrl( Url : String)
13944:     Procedure GLShowCursor( AShow : boolean)
13945:     Procedure GLSetCursorPos( AScreenX, AScreenY : integer)
13946:     Procedure GLGetCursorPos( var point : TGLPoint)
13947:     Function GLGetScreenWidth : integer
13948:     Function GLGetScreenHeight : integer
13949:     Function GLGetTickCount : int64
13950:     function RemoveSpaces(const str : String) : String;
13951:     TNormalMapSpace, '( nmsObject, nmsTangent )
13952:     Procedure CalcObjectSpaceLightVectors(Light:TAffineVector;Vertices TAffineVectorList;Colors:TVectorList)
13953:     Procedure SetupTangentSpace( Vertices, Normals, TexCoords, Tangents, BiNormals : TAffineVectorList)
13954:     Procedure CalcTangentSpaceLightVectors( Light : TAffineVector; Vertices, Normals, Tangents, BiNormals :
        TAffineVectorList; Colors : TVectorList)
13955:     Function CreateObjectSpaceNormalMap( Width,Height:Integer;HiNormals,
        HiTexCoords:TAffineVectorList):TGLBitmap

```

```

13956: Function CreateTangentSpaceNormalMap( Width, Height : Integer; HiNormals, HiTexCoords, LoNormals,
LoTexCoords, Tangents, BiNormals : TAffineVectorList) : TGLBitmap
13957: end;
13958:
13959: procedure SIRegister_GLStarRecord(CL: TPSPascalCompiler);
13960: begin
13961:   TGLStarRecord', 'record RA: Word; DEC: SmallInt; BVColorIndex: Byte; VMagnitude: Byte; end
13962:   // PGLStarRecord', '^TGLStarRecord // will not work
13963: Function StarRecordPositionZUp( const starRecord : TGLStarRecord) : TAffineVector
13964: Function StarRecordPositionYUp( const starRecord : TGLStarRecord) : TAffineVector
13965: Function StarRecordFromColor( const starRecord : TGLStarRecord; bias : Single) : TVector
13966: end;
13967:
13968:
13969: procedure SIRegister_GeometryBB(CL: TPSPascalCompiler);
13970: begin
13971:   TAABB', 'record min : TAffineVector; max : TAffineVector; end
13972:   //PAABB', '^TAABB // will not work
13973:   TBSphere', 'record Center : TAffineVector; Radius : single; end
13974:   TClipRect', 'record Left : Single; Top:Single; Right:Single; Bottom : Single; end
13975:   TSpaceContains', '(scNoOverlap, scContainsFully, scContainsPartially )
13976: Function AddBB( var c1 : THmgBoundingBox; const c2 : THmgBoundingBox) : THmgBoundingBox
13977: Procedure AddAABB( var aabb : TAABB; const aabb1 : TAABB)
13978: Procedure SetBB( var c : THmgBoundingBox; const v : TVector)
13979: Procedure SetAABB( var bb : TAABB; const v : TVector)
13980: Procedure BBTransform( var c : THmgBoundingBox; const m : TMatrix)
13981: Procedure AABBTransform( var bb : TAABB; const m : TMatrix)
13982: Procedure AABBScale( var bb : TAABB; const v : TAffineVector)
13983: Function BBMinX( const c : THmgBoundingBox) : Single
13984: Function BBMaxX( const c : THmgBoundingBox) : Single
13985: Function BBMinY( const c : THmgBoundingBox) : Single
13986: Function BBMaxY( const c : THmgBoundingBox) : Single
13987: Function BBMinZ( const c : THmgBoundingBox) : Single
13988: Function BBMaxZ( const c : THmgBoundingBox) : Single
13989: Procedure AABBInclude( var bb : TAABB; const p : TAffineVector)
13990: Procedure AABBFromSweep( var SweepAABB : TAABB; const Start, Dest : TVector; const Radius : Single)
13991: Function AABBIntersection( const aabb1, aabb2 : TAABB) : TAABB
13992: Function BBToAABB( const aabb : THmgBoundingBox) : TAABB
13993: Function AABBToBB( const anAABB : TAABB) : THmgBoundingBox;
13994: Function AABBToBB1( const anAABB : TAABB; const m : TMatrix) : THmgBoundingBox;
13995: Procedure OffsetAABB( var aabb : TAABB; const delta : TAffineVector);
13996: Procedure OffsetAABB1( var aabb : TAABB; const delta : TVector);
13997: Function IntersectAABBs( const aabb1, aabb2 : TAABB; const m1To2, m2To1 : TMatrix) : Boolean;
13998: Function IntersectAABBsAbsoluteXY( const aabb1, aabb2 : TAABB) : Boolean
13999: Function IntersectAABBsAbsoluteXZ( const aabb1, aabb2 : TAABB) : Boolean
14000: Function IntersectAABBsAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14001: Function AABBFFitsInAABBAbsolute( const aabb1, aabb2 : TAABB) : Boolean
14002: Function PointInAABB( const p : TAffineVector; const aabb : TAABB) : Boolean;
14003: Function PointInAABB1( const p : TVector; const aabb : TAABB) : Boolean;
14004: Function PlaneIntersectAABB( Normal : TAffineVector; d : single; aabb : TAABB) : boolean
14005: Function TriangleIntersectAABB( const aabb : TAABB; v1, v2, v3 : TAffineVector) : boolean
14006: Procedure ExtractAABBCorners( const AABB : TAABB; var AABBCorners : TAABBCorners)
14007: Procedure AABBToBSphere( const AABB : TAABB; var BSphere : TBSphere)
14008: Procedure BSphereToAABB( const BSphere : TBSphere; var AABB : TAABB);
14009: Function BSphereToAABB1( const center : TAffineVector; radius : Single) : TAABB;
14010: Function BSphereToAABB2( const center : TVector; radius : Single) : TAABB;
14011: Function AABBContainsAABB( const mainAABB, testAABB : TAABB) : TSpaceContains
14012: Function BSphereContainsAABB( const mainBSphere : TBSphere; const testAABB : TAABB) : TSpaceContains
14013: Function BSphereContainsBSphere( const mainBSphere, testBSphere : TBSphere) : TSpaceContains
14014: Function AABBContainsBSphere( const mainAABB : TAABB; const testBSphere : TBSphere) : TSpaceContains
14015: Function PlaneContainsBSphere( const Location,Normal:TAffineVector;const
testBSphere:TBSphere):TSpaceContains
14016: Function FrustumContainsBSphere( const Frustum : TFrustum; const testBSphere : TBSphere) : TSpaceContains
14017: Function FrustumContainsAABB( const Frustum : TFrustum; const testAABB : TAABB) : TSpaceContains
14018: Function ClipToAABB( const v : TAffineVector; const AABB : TAABB) : TAffineVector
14019: Function BSphereIntersectsBSphere( const mainBSphere, testBSphere : TBSphere) : boolean
14020: Procedure IncludeInClipRect( var clipRect : TClipRect; x, y : Single)
14021: Function AABBToClipRect(const aabb:TAABB;modelViewProjection:TMatrix;viewportSizeX,
viewportSizeY:Int):TClipRect
14022: end;
14023:
14024: procedure SIRegister_GeometryCoordinates(CL: TPSPascalCompiler);
14025: begin
14026:   Procedure Cylindrical_Cartesian( const r, theta, z1 : single; var x, y, z : single);
14027:   Procedure Cylindrical_Cartesian1( const r, theta, z1 : double; var x, y, z : double);
14028:   Procedure Cylindrical_Cartesian2( const r,theta,z1 : single; var x, y, z : single; var ierr : integer);
14029:   Procedure Cylindrical_Cartesian3( const r,theta,z1 : double; var x, y, z : double; var ierr : integer);
14030:   Procedure Cartesian_Cylindrical( const x, y, z1 : single; var r, theta, z : single);
14031:   Procedure Cartesian_Cylindrical1( const x, y, z1 : double; var r, theta, z : double);
14032:   Procedure Spherical_Cartesian( const r, theta, phi : single; var x, y, z : single);
14033:   Procedure Spherical_Cartesian1( const r, theta, phi : double; var x, y, z : double);
14034:   Procedure Spherical_Cartesian2( const r,theta, phi : single; var x, y, z : single; var ierr : integer);
14035:   Procedure Spherical_Cartesian3( const r,theta, phi : double; var x, y, z : double; var ierr : integer);
14036:   Procedure Cartesian_Spherical( const x, y, z : single; var r, theta, phi : single);
14037:   Procedure Cartesian_Spherical1( const v : TAffineVector; var r, theta, phi : Single);
14038:   Procedure Cartesian_Spherical2( const x, y, z : double; var r, theta, phi : double);
14039:   Procedure ProlateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14040:   Procedure ProlateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14041:   Procedure ProlateSpheroidal_Cartesian2(const xi,eta,phi,a:single;var x,y,z:single;var ierr: integer);

```

```

14042: Procedure ProlateSpheroidal_Cartesian3(const xi,eta,phi,a:double;var x,y,z:double;var ierr: integer);
14043: Procedure OblateSpheroidal_Cartesian( const xi, eta, phi, a : single; var x, y, z : single);
14044: Procedure OblateSpheroidal_Cartesian1( const xi, eta, phi, a : double; var x, y, z : double);
14045: Procedure OblateSpheroidal_Cartesian2(const xi,eta,phi,a:single; var x,y,z: single;var ierr:integer);
14046: Procedure OblateSpheroidal_Cartesian3( const xi,eta,phi,a:double; var x,y,z:double;var ierr:integer);
14047: Procedure BipolarCylindrical_Cartesian( const u, v, z1, a : single; var x, y, z : single);
14048: Procedure BipolarCylindrical_Cartesian1(const u, v, z1, a : double; var x, y, z : double);
14049: Procedure BipolarCylindrical_Cartesian2(const u,v,z1,a: single;var x,y,z:single; var ierr: integer);
14050: Procedure BipolarCylindrical_Cartesian3(const u,v,z1,a: double;var x,y,z:double; var ierr: integer);
14051: end;
14052:
14053: procedure SIRegister_VectorGeometry(CL: TPSPascalCompiler);
14054: begin
14055:   'EPSILON','Single').setExtended( 1e-40);
14056:   'EPSILON2','Single').setExtended( 1e-30); }
14057: TRenderContextClippingInfo, 'record origin : TVector; clippingD'
14058:   + 'irection : TVector; viewPortRadius : Single; farClippingDistance:Single;frustum:TFrustum; end
14059: THmgPlane', 'TVector
14060:   TDoubleHmgPlane', 'THomogeneousDblVector
14061: {TTransType', '( ttScaleX, ttScaleY, ttScaleZ, ttShearXY, ttShear'
14062:   + 'XZ, ttShearYZ, ttRotateX, ttRotateY, ttRotateZ, ttTranslateX, ttTranslateY'
14063:   + ', ttTranslateZ, ttPerspectiveX, ttPerspectiveY, ttPerspectiveZ, ttPerspectiveW }}
14064: TSingleArray', 'array of Single
14065: TTransformations', 'array [0..15] of Single)
14066: TPackedRotationMatrix', 'array [0..2] of Smallint)
14067: TVertex', 'TAffineVector
14068: //TVectorGL', 'THomogeneousFltVector
14069: //TMatrixGL', 'THomogeneousFltMatrix
14070: // TPackedRotationMatrix = array [0..2] of SmallInt;
14071: Function glTexPointMake( const s, t : Single) : TTexPoint
14072: Function glAffineVectorMake( const x, y, z : Single) : TAffineVector;
14073: Function glAffineVectorMakel( const v : TVectorGL) : TAffineVector;
14074: Procedure glSetAffineVector( var v : TAffineVector; const x, y, z : Single);
14075: Procedure glSetVector( var v : TAffineVector; const x, y, z : Single);
14076: Procedure glSetVector1( var v : TAffineVector; const vSrc : TVectorGL);
14077: Procedure glSetVector2( var v : TAffineVector; const vSrc : TAffineVector);
14078: Procedure glSetVector3( var v : TAffineDblVector; const vSrc : TAffineVector);
14079: Procedure glSetVector4( var v : TAffineDblVector; const vSrc : TVectorGL);
14080: Function glVectorMake( const v : TAffineVector; w : Single) : TVectorGL;
14081: Function glVectorMakel( const x, y, z : Single; w : Single) : TVectorGL;
14082: Function glPointMake( const x, y, z : Single) : TVectorGL;
14083: Function glPointMakel( const v : TAffineVector) : TVectorGL;
14084: Function glPointMake2( const v : TVectorGL) : TVectorGL;
14085: Procedure glSetVector5( var v : TVectorGL; const x, y, z : Single; w : Single);
14086: Procedure glSetVector6( var v : TVectorGL; const av : TAffineVector; w : Single);
14087: Procedure glglSetVector7( var v : TVectorGL; const vSrc : TVectorGL);
14088: Procedure glMakePoint( var v : TVectorGL; const x, y, z : Single);
14089: Procedure glMakePoint1( var v : TVectorGL; const av : TAffineVector);
14090: Procedure glMakePoint2( var v : TVectorGL; const av : TVectorGL);
14091: Procedure glMakeVector( var v : TAffineVector; const x, y, z : Single);
14092: Procedure glMakeVector1( var v : TVectorGL; const x, y, z : Single);
14093: Procedure glMakeVector2( var v : TVectorGL; const av : TAffineVector);
14094: Procedure glMakeVector3( var v : TVectorGL; const av : TVectorGL);
14095: Procedure glRstVector( var v : TAffineVector);
14096: Procedure glRstVector1( var v : TVectorGL);
14097: Function glVectorAdd( const v1, v2 : TAffineVector) : TAffineVector;
14098: Procedure glVectorAdd1( const v1, v2 : TAffineVector; var vr : TAffineVector);
14099: //Procedure VectorAdd2( const v1, v2 : TAffineVector; vr : PAffineVector);
14100: Function glVectorAdd3( const v1, v2 : TVectorGL) : TVectorGL;
14101: Procedure glVectorAdd4( const v1, v2 : TVectorGL; var vr : TVectorGL);
14102: Function glVectorAdd5( const v : TAffineVector; const f : Single) : TAffineVector;
14103: Function glVectorAdd6( const v : TVectorGL; const f : Single) : TVectorGL;
14104: Procedure glAddVector7( var v1 : TAffineVector; const v2 : TAffineVector);
14105: Procedure glAddVector8( var v1 : TAffineVector; const v2 : TVectorGL);
14106: Procedure glAddVector9( var v1 : TVectorGL; const v2 : TVectorGL);
14107: Procedure glAddVector10( var v : TAffineVector; const f : Single);
14108: Procedure glAddVector11( var v : TVectorGL; const f : Single);
14109: //Procedure TexPointArrayAdd(const src:PTexPointArray;const delta:TTexPoint;const
nb:Int;dest:PTexPointArray);
14110: //Procedure TexPointArrayScaleAndAdd(const src:PTexPointArray;const delta:TTexPoint;const
nb:Integer;const scale: TTexPoint; dest : PTexPointArray);
14111: //Procedure VectorArrayAdd(const src:PAffineVectorArray;const delta:TAffineVector;const nb:Integer;dest:
PAffineVectorArray);
14112: Function glVectorSubtract( const V1, V2 : TAffineVector) : TAffineVector;
14113: Procedure glVectorSubtract1( const v1, v2 : TAffineVector; var result : TAffineVector);
14114: Procedure glVectorSubtract2( const v1, v2 : TAffineVector; var result : TVectorGL);
14115: Procedure glVectorSubtract3( const v1 : TVectorGL; v2 : TAffineVector; var result : TVectorGL);
14116: Function glVectorSubtract4( const V1, V2 : TVectorGL) : TVectorGL;
14117: Procedure glVectorSubtract5( const v1, v2 : TVectorGL; var result : TVectorGL);
14118: Procedure glVectorSubtract6( const v1, v2 : TVectorGL; var result : TAffineVector);
14119: Function glVectorSubtract7( const v1 : TAffineVector; delta : Single) : TAffineVector;
14120: Function glVectorSubtract8( const v1 : TVectorGL; delta : Single) : TVectorGL;
14121: Procedure glSubtractVector9( var V1 : TAffineVector; const V2 : TAffineVector);
14122: Procedure glSubtractVector10( var V1 : TVectorGL; const V2 : TVectorGL);
14123: Procedure glCombineVector( var vr : TAffineVector; const v : TAffineVector; var f : Single);
14124: //Procedure CombineVector1( var vr : TAffineVector; const v : TAffineVector; pf : PFloat);
14125: Function glTexPointCombine( const t1, t2 : TTexPoint; f1, f2 : Single) : TTexPoint
14126: Function glVectorCombine2( const V1, V2 : TAffineVector; const F1, F2 : Single) : TAffineVector;
14127: Function glVectorCombine33( const V1, V2, V3 : TAffineVector; const F1, F2, F3 : Single) : TAffineVector;

```



```

14128: Procedure glVectorCombine34(const V1,V2,V3:TAffineVector; const F1,F2,F3: Single; var vr : TAffineVector);
14129: Procedure glCombineVector5( var vr : TVectorGL; const v : TVectorGL; var f : Single);
14130: Procedure glCombineVector6( var vr : TVectorGL; const v : TAffineVector; var f : Single);
14131: Function glVectorCombine7( const V1, V2 : TVectorGL; const F1, F2 : Single) : TVectorGL;
14132: Function glVectorCombine8( const V1 : TVectorGL;const V2: TAffineVector; const F1,F2:Single): TVectorGL;
14133: Procedure glVectorCombine9(const V1:TVectorGL;const V2:TAffineVector;const F1,F2:Single;var vr:TVectorGL);
14134: Procedure glVectorCombine10( const V1, V2 : TVectorGL; const F1, F2 : Single; var vr : TVectorGL);
14135: Procedure glVectorCombine11( const V1, V2 : TVectorGL; const F2 : Single; var vr : TVectorGL);
14136: Function glVectorCombine3( const V1, V2, V3 : TVectorGL; const F1, F2, F3 : Single) : TVectorGL;
14137: Procedure glVectorCombine31( const V1, V2, V3 : TVectorGL; const F1, F2, F3:Single; var vr : TVectorGL);
14138: Function glVectorDotProduct( const V1, V2 : TAffineVector) : Single;
14139: Function glVectorDotProduct1( const V1, V2 : TVectorGL) : Single;
14140: Function glVectorDotProduct2( const V1 : TVectorGL; const V2 : TAffineVector) : Single;
14141: Function glPointProject( const p, origin, direction : TAffineVector) : Single;
14142: Function glPointProject1( const p, origin, direction : TVectorGL) : Single;
14143: Function glVectorCrossProduct( const V1, V2 : TAffineVector) : TAffineVector;
14144: Function glVectorCrossProduct1( const V1, V2 : TVectorGL) : TVectorGL;
14145: Procedure glVectorCrossProduct2( const v1, v2 : TVectorGL; var vr : TVectorGL);
14146: Procedure glVectorCrossProduct3( const v1, v2 : TAffineVector; var vr : TVectorGL);
14147: Procedure glVectorCrossProduct4( const v1, v2 : TVectorGL; var vr : TAffineVector);
14148: Procedure glVectorCrossProduct5( const v1, v2 : TAffineVector; var vr : TAffineVector);
14149: Function glLerp( const start, stop, t : Single) : Single
14150: Function glAngleLerp( start, stop, t : Single) : Single
14151: Function glDistanceBetweenAngles( angle1, angle2 : Single) : Single
14152: Function glTexPointLerp( const t1, t2 : TTexPoint; t : Single) : TTexPoint;
14153: Function glVectorLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14154: Procedure glVectorLerp1( const v1, v2 : TAffineVector; t : Single; var vr : TAffineVector);
14155: Function glVectorLerp2( const v1, v2 : TVectorGL; t : Single) : TVectorGL;
14156: Procedure glVectorLerp3( const v1, v2 : TVectorGL; t : Single; var vr : TVectorGL);
14157: Function glVectorAngleLerp( const v1, v2 : TAffineVector; t : Single) : TAffineVector;
14158: Function glVectorAngleCombine( const v1, v2 : TAffineVector; f : Single) : TAffineVector;
14159: // Procedure VectorArrayLerp( const src1, src2:PVectorArray; t:Single; n:Integer; dest:PVectorArray);
14160: // Procedure VectorArrayLerp1( const src1, src2 : PAffineVectorArray; t : Single; n : Integer; dest :
PAffineVectorArray);
14161: Function glVectorLength( const x, y : Single) : Single;
14162: Function glVectorLength1( const x, y, z : Single) : Single;
14163: Function glVectorLength2( const v : TAffineVector) : Single;
14164: Function glVectorLength3( const v : TVectorGL) : Single;
14165: Function glVectorLength4( const v : array of Single) : Single;
14166: Function glVectorNorm( const x, y : Single) : Single;
14167: Function glVectorNorm1( const v : TAffineVector) : Single;
14168: Function glVectorNorm2( const v : TVectorGL) : Single;
14169: Function glVectorNorm3( var V : array of Single) : Single;
14170: Procedure glNormalizeVector( var v : TAffineVector);
14171: Procedure glNormalizeVector1( var v : TVectorGL);
14172: Function glVectorNormalize( const v : TAffineVector) : TAffineVector;
14173: Function glVectorNormalize1( const v : TVectorGL) : TVectorGL;
14174: // Procedure NormalizeVectorArray( list : PAffineVectorArray; n : Integer);
14175: Function glVectorAngleCosine( const V1, V2 : TAffineVector) : Single
14176: Function glVectorNegate( const v : TAffineVector) : TAffineVector;
14177: Function glVectorNegate1( const v : TVectorGL) : TVectorGL;
14178: Procedure glNegateVector( var V : TAffineVector);
14179: Procedure glNegateVector2( var V : TVectorGL);
14180: Procedure glNegateVector3( var V : array of Single);
14181: Procedure glScaleVector( var v : TAffineVector; factor : Single);
14182: Procedure glScaleVector1( var v : TAffineVector; const factor : TAffineVector);
14183: Procedure glScaleVector2( var v : TVectorGL; factor : Single);
14184: Procedure glScaleVector3( var v : TVectorGL; const factor : TVectorGL);
14185: Function glVectorScale( const v : TAffineVector; factor : Single) : TAffineVector;
14186: Procedure glVectorScale1( const v : TAffineVector; factor : Single; var vr : TAffineVector);
14187: Function glVectorScale2( const v : TVectorGL; factor : Single) : TVectorGL;
14188: Procedure glVectorScale3( const v : TVectorGL; factor : Single; var vr : TVectorGL);
14189: Procedure glVectorScale4( const v : TVectorGL; factor : Single; var vr : TAffineVector);
14190: Procedure glDivideVector( var v : TVectorGL; const divider : TVectorGL);
14191: Function glVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14192: Function glVectorEquals1( const V1, V2 : TAffineVector) : Boolean;
14193: Function glAffineVectorEquals( const V1, V2 : TVectorGL) : Boolean;
14194: Function glVectorIsNull( const v : TVectorGL) : Boolean;
14195: Function glVectorIsNull1( const v : TAffineVector) : Boolean;
14196: Function glVectorSpacing( const v1, v2 : TTexPoint) : Single;
14197: Function glVectorSpacing1( const v1, v2 : TAffineVector) : Single;
14198: Function glVectorSpacing2( const v1, v2 : TVectorGL) : Single;
14199: Function glVectorDistance( const v1, v2 : TAffineVector) : Single;
14200: Function glVectorDistance1( const v1, v2 : TVectorGL) : Single;
14201: Function glVectorDistance2( const v1, v2 : TAffineVector) : Single;
14202: Function glVectorDistance21( const v1, v2 : TVectorGL) : Single;
14203: Function glVectorPerpendicular( const V, N : TAffineVector) : TAffineVector
14204: Function glVectorReflect( const V, N : TAffineVector) : TAffineVector
14205: Procedure glRotateVector( var vector : TVectorGL; const axis : TAffineVector; angle : Single);
14206: Procedure glRotateVector1( var vector : TVectorGL; const axis : TVectorGL; angle : Single);
14207: Procedure glRotateVectorAroundY( var v : TAffineVector; alpha : Single)
14208: Function glVectorRotateAroundX( const v : TAffineVector; alpha : Single) : TAffineVector;
14209: Function glVectorRotateAroundY( const v : TAffineVector; alpha : Single) : TAffineVector;
14210: Procedure glVectorRotateAroundY1(const v : TAffineVector; alpha : Single; var vr : TAffineVector);
14211: Function glVectorRotateAroundZ( const v : TAffineVector; alpha : Single) : TAffineVector;
14212: Procedure glAbsVector( var v : TVectorGL);
14213: Procedure glAbsVector1( var v : TAffineVector);
14214: Function glVectorAbs( const v : TVectorGL) : TVectorGL;
14215: Function glVectorAbs1( const v : TAffineVector) : TAffineVector;

```

```

14216: Procedure glSetMatrix( var dest : THomogeneousDblMatrix; const src : TMatrixGL);
14217: Procedure glSetMatrix1( var dest : TAffineMatrix; const src : TMatrixGL);
14218: Procedure glSetMatrix2( var dest : TMatrixGL; const src : TAffineMatrix);
14219: Procedure glSetMatrixRow( var dest : TMatrixGL; rowNb : Integer; const aRow : TVectorGL);
14220: Function glCreateScaleMatrix( const v : TAffineVector) : TMatrixGL;
14221: Function glCreateScaleMatrix1( const v : TVectorGL) : TMatrixGL;
14222: Function glCreateTranslationMatrix( const V : TAffineVector) : TMatrixGL;
14223: Function glCreateTranslationMatrix1( const V : TVectorGL) : TMatrixGL;
14224: Function glCreateScaleAndTranslationMatrix( const scale, offset : TVectorGL) : TMatrixGL;
14225: Function glCreateRotationMatrixX( const sine, cosine : Single) : TMatrixGL;
14226: Function glCreateRotationMatrixX1( const angle : Single) : TMatrixGL;
14227: Function glCreateRotationMatrixY( const sine, cosine : Single) : TMatrixGL;
14228: Function glCreateRotationMatrixY1( const angle : Single) : TMatrixGL;
14229: Function glCreateRotationMatrixZ( const sine, cosine : Single) : TMatrixGL;
14230: Function glCreateRotationMatrixZ1( const angle : Single) : TMatrixGL;
14231: Function glCreateRotationMatrix( const anAxis : TAffineVector; angle : Single) : TMatrixGL;
14232: Function glCreateRotationMatrix1( const anAxis : TVectorGL; angle : Single) : TMatrixGL;
14233: Function glCreateAffineRotationMatrix( const anAxis : TAffineVector; angle:Single):TAffineMatrix
14234: Function glMatrixMultiply( const M1, M2 : TAffineMatrix) : TAffineMatrix;
14235: Function glMatrixMultiply1( const M1, M2 : TMatrixGL) : TMatrixGL;
14236: Procedure glMatrixMultiply2( const M1, M2 : TMatrixGL; var MResult : TMatrixGL);
14237: Function glVectorTransform( const V : TVectorGL; const M : TMatrixGL) : TVectorGL;
14238: Function glVectorTransform1( const V : TVectorGL; const M : TAffineMatrix) : TVectorGL;
14239: Function glVectorTransform2( const V : TAffineVector; const M : TMatrixGL) : TAffineVector;
14240: Function glVectorTransform3( const V : TAffineVector; const M : TAffineMatrix) : TAffineVector;
14241: Function glMatrixDeterminant( const M : TAffineMatrix) : Single;
14242: Function glMatrixDeterminant1( const M : TMatrixGL) : Single;
14243: Procedure glAdjointMatrix( var M : TMatrixGL);
14244: Procedure glAdjointMatrix1( var M : TAffineMatrix);
14245: Procedure glScaleMatrix( var M : TAffineMatrix; const factor : Single);
14246: Procedure glScaleMatrix1( var M : TMatrixGL; const factor : Single);
14247: Procedure glTranslateMatrix( var M : TMatrixGL; const v : TAffineVector);
14248: Procedure glTranslateMatrix1( var M : TMatrixGL; const v : TVectorGL);
14249: Procedure glNormalizeMatrix( var M : TMatrixGL);
14250: Procedure glTransposeMatrix( var M : TAffineMatrix);
14251: Procedure glTransposeMatrix1( var M : TMatrixGL);
14252: Procedure glInvertMatrix( var M : TMatrixGL);
14253: Procedure glInvertMatrix1( var M : TAffineMatrix);
14254: Function glAnglePreservingMatrixInvert( const mat : TMatrixGL) : TMatrixGL
14255: Function glMatrixDecompose( const M : TMatrixGL; var Tran : TTransformations) : Boolean
14256: Function glPlaneMake( const p1, p2, p3 : TAffineVector) : THmgPlane;
14257: Function glPlaneMake1( const p1, p2, p3 : TVectorGL) : THmgPlane;
14258: Function glPlaneMake2( const point, normal : TAffineVector) : THmgPlane;
14259: Function glPlaneMake3( const point, normal : TVectorGL) : THmgPlane;
14260: Procedure glSetPlane( var dest : TDoubleHmgPlane; const src : THmgPlane)
14261: Procedure glNormalizePlane( var plane : THmgPlane)
14262: Function glPlaneEvaluatePoint( const plane : THmgPlane; const point : TAffineVector) : Single;
14263: Function glPlaneEvaluatePoint1( const plane : THmgPlane; const point : TVectorGL) : Single;
14264: Function glCalcPlaneNormal( const p1, p2, p3 : TAffineVector) : TAffineVector;
14265: Procedure glCalcPlaneNormal1( const p1, p2, p3 : TAffineVector; var vr : TAffineVector);
14266: Procedure glCalcPlaneNormal2( const p1, p2, p3 : TVectorGL; var vr : TAffineVector);
14267: Function glPointIsInHalfSpace( const point, planePoint, planeNormal : TVectorGL) : Boolean;
14268: Function glPointIsInHalfSpace1( const point, planePoint, planeNormal : TAffineVector) : Boolean;
14269: Function glPointPlaneDistance( const point, planePoint, planeNormal : TVectorGL) : Single;
14270: Function glPointPlaneDistance1( const point, planePoint, planeNormal : TAffineVector) : Single;
14271: Function glPointSegmentClosestPoint( const point, segmentStart, segmentStop:TAffineVector):TAffineVector
14272: Function glPointSegmentDistance( const point, segmentStart, segmentStop : TAffineVector) : single
14273: Function glPointLineClosestPoint( const point, linePoint, lineDirection : TAffineVector) : TAffineVector
14274: Function glPointLineDistance( const point, linePoint, lineDirection : TAffineVector) : Single
14275: Procedure SgsegmentSegmentClosestPoint( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector; var
Segment0Closest, Segment1Closest : TAffineVector)
14276: Function glSegmentSegmentDistance( const S0Start, S0Stop, S1Start, S1Stop : TAffineVector) : single
14277: TEulerOrder', '( eulXYZ, eulXZY, eulYXZ, eulYZX, eulZXY, eulZYX)
14278: Function glQuaternionMake( const Imag : array of Single; Real : Single) : TQuaternion
14279: Function glQuaternionConjugate( const Q : TQuaternion) : TQuaternion
14280: Function glQuaternionMagnitude( const Q : TQuaternion) : Single
14281: Procedure glNormalizeQuaternion( var Q : TQuaternion)
14282: Function glQuaternionFromPoints( const V1, V2 : TAffineVector) : TQuaternion
14283: Procedure glQuaternionToPoints( const Q : TQuaternion; var ArcFrom, ArcTo : TAffineVector)
14284: Function glQuaternionFromMatrix( const mat : TMatrixGL) : TQuaternion
14285: Function glQuaternionToMatrix( quat : TQuaternion) : TMatrixGL
14286: Function glQuaternionToAffineMatrix( quat : TQuaternion) : TAffineMatrix
14287: Function glQuaternionFromAngleAxis( const angle : Single; const axis : TAffineVector) : TQuaternion
14288: Function glQuaternionFromRollPitchYaw( const r, p, y : Single) : TQuaternion
14289: Function glQuaternionFromEuler( const x, y, z : Single; eulerOrder : TEulerOrder) : TQuaternion
14290: Function glQuaternionMultiply( const qL, qR : TQuaternion) : TQuaternion
14291: Function glQuaternionSlerp( const QStart, QEnd : TQuaternion; Spin : Integer; t : Single) : TQuaternion;
14292: Function glQuaternionSlerp1( const source, dest : TQuaternion; const t : Single) : TQuaternion;
14293: Function glLnXP1( X : Extended) : Extended
14294: Function glLog10( X : Extended) : Extended
14295: Function glLog2( X : Extended) : Extended;
14296: Function glLog21( X : Single) : Single;
14297: Function glLogN( Base, X : Extended) : Extended
14298: Function glIntPower( Base : Extended; Exponent : Integer) : Extended
14299: Function glPower( const Base, Exponent : Single) : Single;
14300: Function glPower1( Base : Single; Exponent : Integer) : Single;
14301: Function glDegToRad( const Degrees : Extended) : Extended;
14302: Function glDegToRad1( const Degrees : Single) : Single;
14303: Function glRadToDeg( const Radians : Extended) : Extended;

```

```

14304: Function glRadToDeg1( const Radians : Single) : Single;
14305: Function glNormalizeAngle( angle : Single) : Single
14306: Function glNormalizeDegAngle( angle : Single) : Single
14307: Procedure glSinCos( const Theta : Extended; var Sin, Cos : Extended);
14308: Procedure glSinCos1( const Theta : Double; var Sin, Cos : Double);
14309: Procedure glSinCos( const Theta : Single; var Sin, Cos : Single);
14310: Procedure glSinCos1( const theta, radius : Double; var Sin, Cos : Extended);
14311: Procedure glSinCos2( const theta, radius : Double; var Sin, Cos : Double);
14312: Procedure glSinCos3( const theta, radius : Single; var Sin, Cos : Single);
14313: Procedure glPrepareSinCosCache( var s, c : array of Single; startAngle, stopAngle : Single)
14314: Function glArcCos( const X : Extended) : Extended;
14315: Function glArcCos1( const x : Single) : Single;
14316: Function glArcSin( const X : Extended) : Extended;
14317: Function glArcSin1( const X : Single) : Single;
14318: Function glArcTan21( const Y, X : Extended) : Extended;
14319: Function glArcTan21( const Y, X : Single) : Single;
14320: Function glFastArcTan2( y, x : Single) : Single
14321: Function glTan( const X : Extended) : Extended;
14322: Function glTan1( const X : Single) : Single;
14323: Function glCoTan( const X : Extended) : Extended;
14324: Function glCoTan1( const X : Single) : Single;
14325: Function glSinh( const x : Single) : Single;
14326: Function glSinh1( const x : Double) : Double;
14327: Function glCosh( const x : Single) : Single;
14328: Function glCosh1( const x : Double) : Double;
14329: Function glRSqrt( v : Single) : Single
14330: Function glRLength( x, y : Single) : Single
14331: Function glISqrt( i : Integer) : Integer
14332: Function glILength( x, y : Integer) : Integer;
14333: Function glILength1( x, y, z : Integer) : Integer;
14334: Procedure glRegisterBasedExp
14335: Procedure glRandomPointOnSphere( var p : TAffineVector)
14336: Function glRoundInt( v : Single) : Single;
14337: Function glRoundInt1( v : Extended) : Extended;
14338: Function glTrunc( v : Single) : Integer;
14339: Function glTrunc64( v : Extended) : Int64;
14340: Function glInt( v : Single) : Single;
14341: Function glInt1( v : Extended) : Extended;
14342: Function glFrac( v : Single) : Single;
14343: Function glFrac1( v : Extended) : Extended;
14344: Function glRound( v : Single) : Integer;
14345: Function glRound64( v : Single) : Int64;
14346: Function glRound641( v : Extended) : Int64;
14347: Function glTrunc( X : Extended) : Int64
14348: Function glRound( X : Extended) : Int64
14349: Function glFrac( X : Extended) : Extended
14350: Function glCeil( v : Single) : Integer;
14351: Function glCeil64( v : Extended) : Int64;
14352: Function glFloor( v : Single) : Integer;
14353: Function glFloor64( v : Extended) : Int64;
14354: Function glScaleAndRound( i : Integer; var s : Single) : Integer
14355: Function glSign( x : Single) : Integer
14356: Function glIsInRange( const x, a, b : Single) : Boolean;
14357: Function glIsInRangel( const x, a, b : Double) : Boolean;
14358: Function glIsInCube( const p, d : TAffineVector) : Boolean;
14359: Function glIsInCubel( const p, d : TVectorGL) : Boolean;
14360: //Function MinFloat( values : PSingleArray; nbItems : Integer) : Single;
14361: //Function MinFloat1( values : PDoubleArray; nbItems : Integer) : Double;
14362: //Function MinFloat2( values : PExtendedArray; nbItems : Integer) : Extended;
14363: Function glMinFloat3( const v1, v2 : Single) : Single;
14364: Function glMinFloat4( const v : array of Single) : Single;
14365: Function glMinFloat5( const v1, v2 : Double) : Double;
14366: Function glMinFloat6( const v1, v2 : Extended) : Extended;
14367: Function glMinFloat7( const v1, v2, v3 : Single) : Single;
14368: Function glMinFloat8( const v1, v2, v3 : Double) : Double;
14369: Function glMinFloat9( const v1, v2, v3 : Extended) : Extended;
14370: //Function MaxFloat10( values : PSingleArray; nbItems : Integer) : Single;
14371: //Function MaxFloat( values : PDoubleArray; nbItems : Integer) : Double;
14372: //Function MaxFloat1( values : PExtendedArray; nbItems : Integer) : Extended;
14373: Function glMaxFloat2( const v : array of Single) : Single;
14374: Function glMaxFloat3( const v1, v2 : Single) : Single;
14375: Function glMaxFloat4( const v1, v2 : Double) : Double;
14376: Function glMaxFloat5( const v1, v2 : Extended) : Extended;
14377: Function glMaxFloat6( const v1, v2, v3 : Single) : Single;
14378: Function glMaxFloat7( const v1, v2, v3 : Double) : Double;
14379: Function glMaxFloat8( const v1, v2, v3 : Extended) : Extended;
14380: Function glMinInteger9( const v1, v2 : Integer) : Integer;
14381: Function glMinInteger( const v1, v2 : Cardinal) : Cardinal;
14382: Function glMaxInteger( const v1, v2 : Integer) : Integer;
14383: Function glMaxInteger1( const v1, v2 : Cardinal) : Cardinal;
14384: Function glTriangleArea( const p1, p2, p3 : TAffineVector) : Single;
14385: //Function PolygonArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14386: Function glTriangleSignedArea( const p1, p2, p3 : TAffineVector) : Single;
14387: //Function PolygonSignedArea( const p : PAffineVectorArray; nSides : Integer) : Single;
14388: //Procedure ScaleFloatArray( values : PSingleArray; nb : Integer; var factor : Single);
14389: Procedure glScaleFloatArray( var values : TSingleArray; factor : Single);
14390: //Procedure OffsetFloatArray( values : PSingleArray; nb : Integer; var delta : Single);
14391: Procedure glOffsetFloatArray1( var values : array of Single; delta : Single);
14392: //Procedure OffsetFloatArray2( valuesDest, valuesDelta : PSingleArray; nb : Integer);

```

```

14393: Function glMaxXYZComponent( const v : TVectorGL ) : Single;
14394: Function glMaxXYZComponent1( const v : TAffineVector ) : single;
14395: Function glMinXYZComponent( const v : TVectorGL ) : Single;
14396: Function glMinXYZComponent1( const v : TAffineVector ) : single;
14397: Function glMaxAbsXYZComponent( v : TVectorGL ) : Single
14398: Function glMinAbsXYZComponent( v : TVectorGL ) : Single
14399: Procedure glMaxVector( var v : TVectorGL; const v1 : TVectorGL);
14400: Procedure glMaxVector1( var v : TAffineVector; const v1 : TAffineVector);
14401: Procedure glMinVector( var v : TVectorGL; const v1 : TVectorGL);
14402: Procedure glMinVector1( var v : TAffineVector; const v1 : TAffineVector);
14403: Procedure glSortArrayAscending( var a : array of Extended)
14404: Function glClampValue( const aValue, aMin, aMax : Single ) : Single;
14405: Function glClampValue1( const aValue, aMin : Single ) : Single;
14406: Function glGeometryOptimizationMode : String
14407: Procedure glBeginFPUOnlySection
14408: Procedure glEndFPUOnlySection
14409: Function glConvertRotation( const Angles : TAffineVector ) : TVectorGL
14410: Function glMakeAffineDblVector( var v : array of Double ) : TAffineDblVector
14411: Function glMakeDblVector( var v : array of Double ) : THomogeneousDblVector
14412: Function glVectorAffineDblToFlt( const v : TAffineDblVector ) : TAffineVector
14413: Function glVectorDblToFlt( const v : THomogeneousDblVector ) : THomogeneousVector
14414: Function glVectorAffineFltToDbl( const v : TAffineVector ) : TAffineDblVector
14415: Function glVectorFltToDbl( const v : TVectorGL ) : THomogeneousDblVector
14416: Function glPointInPolygon( var xp, yp : array of Single; x, y : Single ) : Boolean
14417: Procedure glDivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
14418: Function glTurn( const Matrix : TMatrixGL; angle : Single ) : TMatrixGL;
14419: Function glTurn1( const Matrix : TMatrixGL; const MasterUp:TAffineVector;Angle:Single):TMatrixGL;
14420: Function glPitch( const Matrix : TMatrixGL; Angle : Single ) : TMatrixGL;
14421: Function glPitch1( const Matrix:TMatrixGL;const MasterRight:TAffineVector;Angle:Single):TMatrixGL;
14422: Function glRoll( const Matrix: TMatrixGL; Angle : Single ) : TMatrixGL;
14423: Function glRoll1(const Matrix: TMatrixGL;const MasterDirection:TAffineVector;Angle: Single): TMatrixGL;
14424: Function glRayCastMinDistToPoint( const rayStart, rayVector: TVectorGL;const point:TVectorGL):Single
14425: Function glRayCastIntersectsSphere( const rayStart, rayVector : TVectorGL; const
sphereCenter:TVectorGL;const sphereRadius : Single ) : Boolean;
14426: Function glRayCastSphereIntersect( const rayStart, rayVector : TVectorGL; const sphereCenter:TVectorGL;
const sphereRadius : Single; var i1, i2 : TVectorGL ) : Integer;
14427: Function glSphereVisibleRadius( distance, radius : Single ) : Single
14428: Function glExtractFrustumFromModelViewProjection( const modelViewProj : TMatrixGL ) : TFrustum
14429: Function glIsVolumeClipped( const objPos : TVectorGL; const objRadius : Single; const rcci :
TRenderContextClippingInfo ) : Boolean;
14430: Function glIsVolumeClipped1( const objPos : TAffineVector; const objRadius : Single; const rcci :
TRenderContextClippingInfo ) : Boolean;
14431: Function glIsVolumeClipped2(const min,max:TAffineVector;const rcci : TRenderContextClippingInfo ) : Bool;
14432: Function glIsVolumeClipped3(const objPos:TAffineVector;const objRadius:Single;const
Frustum:TFrustum):Bool;
14433: Function glMakeParallelProjectionMatrix( const plane : THmgPlane; const dir : TVectorGL ) : TMatrixGL
14434: Function glMakeShadowMatrix( const planePoint, planeNormal, lightPos : TVectorGL ) : TMatrixGL
14435: Function glMakeReflectionMatrix( const planePoint, planeNormal : TAffineVector ) : TMatrixGL
14436: Function glPackRotationMatrix( const mat : TMatrixGL ) : TPackedRotationMatrix
14437: Function glUnPackRotationMatrix( const packedMatrix : TPackedRotationMatrix ) : TMatrixGL
14438: 'cPI','Single').setExtended( 3.141592654);
14439: 'cPIdiv180','Single').setExtended( 0.017453292);
14440: 'c180divPI','Single').setExtended( 57.29577951);
14441: 'c2PI','Single').setExtended( 6.283185307);
14442: 'cPIdiv2','Single').setExtended( 1.570796326);
14443: 'cPIdiv4','Single').setExtended( 0.785398163);
14444: 'c3PIdiv4','Single').setExtended( 2.35619449);
14445: 'cInv2PI','Single').setExtended( 1 / 6.283185307);
14446: 'cInv360','Single').setExtended( 1 / 360);
14447: 'c180','Single').setExtended( 180);
14448: 'c360','Single').setExtended( 360);
14449: 'cOneHalf','Single').setExtended( 0.5);
14450: 'cLn10','Single').setExtended( 2.302585093);
14451: {'MinSingle','Extended').setExtended( 1.5e-45);
14452: 'MaxSingle','Extended').setExtended( 3.4e+38);
14453: 'MinDouble','Extended').setExtended( 5.0e-324);
14454: 'MaxDouble','Extended').setExtended( 1.7e+308);
14455: 'MinExtended','Extended').setExtended( 3.4e-4932);
14456: 'MaxExtended','Extended').setExtended( 1.1e+4932);
14457: 'MinComp','Extended').setExtended( - 9.223372036854775807e+18);
14458: 'MaxComp','Extended').setExtended( 9.223372036854775807e+18);}
14459: end;
14460:
14461: procedure SIRegister_GLVectorFileObjects(CL: TPSPascalCompiler);
14462: begin
14463:   AddClassN(FindClass('TOBJECT'),'TMeshObjectList
14464:   (FindClass('TOBJECT'),'TFaceGroups
14465:   TMeshAutoCentering', '( macCenterX, macCenterY, macCenterZ, macUseBarycenter )
14466:   TMeshAutoCenterings', 'set of TMeshAutoCentering
14467:   TMeshObjectMode', '( momTriangles, momTriangleStrip, momFaceGroups )
14468:   SIRegister_TBaseMeshObject(CL);
14469:   (FindClass('TOBJECT'),'TSkeletonFrameList
14470:   TSkeletonFrameTransform', '( sftRotation, sftQuaternion )
14471:   SIRegister_TSkeletonFrame(CL);
14472:   SIRegister_TSkeletonFrameList(CL);
14473:   (FindClass('TOBJECT'),'TSkeleton
14474:   (FindClass('TOBJECT'),'TSkeletonBone
14475:   SIRegister_TSkeletonBoneList(CL);
14476:   SIRegister_TSkeletonRootBoneList(CL);

```



```

14477: SIRegister_TSkeletonBone(CL);
14478: (FindClass('TOBJECT'),'TSkeletonColliderList
14479: SIRegister_TSkeletonCollider(CL);
14480: SIRegister_TSkeletonColliderList(CL);
14481: (FindClass('TOBJECT'),'TGLBaseMesh
14482: TBlendedLerpInfo', 'record frameIndex1 : Integer; frameIndex2 : '
14483:   +'Integer; lerpFactor : Single; weight : Single; externalPositions : TAffine'
14484:   +'VectorList; externalRotations : TAffineVectorList; externalQuaternions : T'
14485:   +'QuaternionList; end
14486: SIRegister_TSkeleton(CL);
14487: TMeshObjectRenderingOption', '( moroGroupByMaterial )
14488: TMeshObjectRenderingOptions', 'set of TMeshObjectRenderingOption
14489: SIRegister_TMeshObject(CL);
14490: SIRegister_TMeshObjectList(CL);
14491: //TMeshObjectListClass', 'class of TMeshObjectList
14492: (FindClass('TOBJECT'),'TMeshMorphTargetList
14493: SIRegister_TMeshMorphTarget(CL);
14494: SIRegister_TMeshMorphTargetList(CL);
14495: SIRegister_TMorphableMeshObject(CL);
14496: TVertexBoneWeight', 'record BoneID : Integer; Weight : Single; end
14497: //TVertexBoneWeightArray', '^TVertexBoneWeightArray // will not work
14498: //TVerticesBoneWeights', '^TVerticesBoneWeights // will not work
14499: TVertexBoneWeightDynArray', 'array of TVertexBoneWeight
14500: SIRegister_TSkeletonMeshObject(CL);
14501: SIRegister_TFaceGroup(CL);
14502: TFaceGroupMeshMode', '( fgmTriangles, fgmTriangleStrip, fgmFl'
14503:   +'atTriangles, fgmTriangleFan, fgmQuads )
14504: SIRegister_TFGVertexIndexList(CL);
14505: SIRegister_TFGVertexNormalTexIndexList(CL);
14506: SIRegister_TFGIndexTexCoordList(CL);
14507: SIRegister_TFaceGroups(CL);
14508: TMeshNormalsOrientation', '( mnoDefault, mnoInvert )
14509: SIRegister_TVectorFile(CL);
14510: //TVectorFileClass', 'class of TVectorFile
14511: SIRegister_TGLGLSMVectorFile(CL);
14512: SIRegister_TGLBaseMesh(CL);
14513: SIRegister_TGLFreeForm(CL);
14514: TGLActorOption', '( aoSkeletonNormalizeNormals )
14515: TGLActorOptions', 'set of TGLActorOption
14516: 'cDefaultGLActorOptions', 'LongInt').Value.ts32:= ord(aoSkeletonNormalizeNormals);
14517: (FindClass('TOBJECT'),'TGLActor
14518: TActorAnimationReference', '( aarMorph, aarSkeleton, aarNone )
14519: SIRegister_TActorAnimation(CL);
14520: TActorAnimationName', 'String
14521: SIRegister_TActorAnimations(CL);
14522: SIRegister_TGLBaseAnimationController(CL);
14523: SIRegister_TGLAnimationController(CL);
14524: TActorFrameInterpolation', '( afpNone, afpLinear )
14525: TActorAnimationMode', '( aamNone, aamPlayOnce, aamLoop, aamBounc'
14526:   +'eForward, aamBounceBackward, aamLoopBackward, aamExternal )
14527: SIRegister_TGLActor(CL);
14528: SIRegister_TVectorFileFormat(CL);
14529: SIRegister_TVectorFileFormatsList(CL);
14530: (FindClass('TOBJECT'),'EInvalidVectorFile
14531: Function GetVectorFileFormats : TVectorFileFormatsList
14532: Function VectorFileFormatsFilter : String
14533: Function VectorFileFormatsSaveFilter : String
14534: Function VectorFileFormatExtensionByIndex( index : Integer) : String
14535: Procedure RegisterVectorFileFormat( const aExtension, aDescription : String; aClass : TVectorFileClass)
14536: Procedure UnregisterVectorFileClass( aClass : TVectorFileClass)
14537: end;
14538:
14539: procedure SIRegister_AxCtrls(CL: TPSPascalCompiler);
14540: begin
14541:   'Class_DColorPropPage', 'TGUID').SetString( '{5CFF5D59-5946-11D0-BDEF-00A024D1875C}'
14542:   'Class_DFontPropPage', 'TGUID').SetString( '{5CFF5D5B-5946-11D0-BDEF-00A024D1875C}'
14543:   'Class_DPicturePropPage', 'TGUID').SetString( '{5CFF5D5A-5946-11D0-BDEF-00A024D1875C}'
14544:   'Class_DStringPropPage', 'TGUID').SetString( '{F42D677E-754B-11D0-BDFB-00A024D1875C}'
14545:   SIRegister_ToleStream(CL);
14546:   (FindClass('TOBJECT'),'TConnectionPoints
14547:   TConnectionKind', '( ckSingle, ckMulti )
14548:   SIRegister_TConnectionPoint(CL);
14549:   SIRegister_TConnectionPoints(CL);
14550:   TDefinePropertyPage', 'Procedure ( const GUID : TGUID)
14551:   (FindClass('TOBJECT'),'TActiveXControlFactory
14552:   SIRegister_TActiveXControl(CL);
14553:   //TActiveXControlClass', 'class of TActiveXControl
14554:   SIRegister_TActiveXControlFactory(CL);
14555:   SIRegister_TActiveFormControl(CL);
14556:   SIRegister_TActiveForm(CL);
14557:   //TActiveFormClass', 'class of TActiveForm
14558:   SIRegister_TActiveFormFactory(CL);
14559:   (FindClass('TOBJECT'),'TPropertyPageImpl
14560:   SIRegister_TPropertyPage(CL);
14561:   //TPropertyPageClass', 'class of TPropertyPage
14562:   SIRegister_TPropertyPageImpl(CL);
14563:   SIRegister_TActiveXPropertyPage(CL);
14564:   SIRegister_TActiveXPropertyPageFactory(CL);
14565:   SIRegister_TCustomAdapter(CL);

```

```

14566:   SIRegister_TAdapterNotifier(CL);
14567:   SIRegister_IFontAccess(CL);
14568:   SIRegister_TFontAdapter(CL);
14569:   SIRegister_IPictureAccess(CL);
14570:   SIRegister_TPictureAdapter(CL);
14571:   SIRegister_TOLEGraphic(CL);
14572:   SIRegister_TStringsAdapter(CL);
14573:   SIRegister_TReflectorWindow(CL);
14574:   Procedure EnumDispatchProperties(Dispatch:IDispatch;PropType:TGUID;VtCode:Int;PropList:TStrings);
14575:   Procedure GetOleFont( Font : TFont; var OleFont : IFontDisp)
14576:   Procedure SetOleFont( Font : TFont; OleFont : IFontDisp)
14577:   Procedure GetOlePicture( Picture : TPicture; var OlePicture : IPictureDisp)
14578:   Procedure SetOlePicture( Picture : TPicture; OlePicture : IPictureDisp)
14579:   Procedure GetOleStrings( Strings : TStrings; var OleStrings : IStrings)
14580:   Procedure SetOleStrings( Strings : TStrings; OleStrings : IStrings)
14581:   Function ParkingWindow : HWND
14582: end;
14583:
14584: procedure SIRegister_synaip(CL: TPSPascalCompiler);
14585: begin
14586:   // Tip6Bytes = array [0..15] of Byte;
14587:   {:binary form of IPv6 address (for string conversion routines)}
14588:   // Tip6Words = array [0..7] of Word;
14589:   AddTypeS('TIP6Bytes', 'array [0..15] of Byte;');
14590:   AddTypeS('TIP6Words', 'array [0..7] of Word;');
14591:   AddTypeS('TIPAdr', 'record Oct1 : Byte; Oct2 : Byte; Oct3 : Byte; Oct4 : Byte; end;');
14592:   AddDelphiFunction('Function synaIsIP( const Value : string) : Boolean;');
14593:   Function synaIsIP6( const Value : string) : Boolean;
14594:   Function synaIPToID( Host : string) : AnsiString;
14595:   Function synaStrToIP6( value : string) : TIP6Bytes;
14596:   Function synaIP6ToStr( value : TIP6Bytes) : string;
14597:   Function synaStrToIP( value : string) : integer;
14598:   Function synaIPToStr( value : integer) : string;
14599:   Function synaReverseIP( Value : AnsiString) : AnsiString;
14600:   Function synaReverseIP6( Value : AnsiString) : AnsiString;
14601:   Function synaExpandIP6( Value : AnsiString) : AnsiString;
14602:   Function xStrToIP( const Value : String) : TIPAdr;
14603:   Function xIPToStr( const Adresse : TIPAdr) : String;
14604:   Function IPToCardinal( const Adresse : TIPAdr) : Cardinal;
14605:   Function CardinalToIP( const Value : Cardinal) : TIPAdr;
14606:
14607: end;
14608:
14609: procedure SIRegister_synacode(CL: TPSPascalCompiler);
14610: begin
14611:   AddTypeS('TSpecials', 'set of Char;');
14612:   Const ('SpecialChar', 'TSpecials').SetSet( '=( ) [ < > ; , @ / ? \ _ ');
14613:   Const ('URLFullSpecialChar', 'TSpecials').SetSet( ' ; / ? : @ = & # + ');
14614:   Const ('TableBase64', 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=');
14615:   Const ('TableBase64mod', 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+,';
14616:   Const ('TableUU'('!'#$%&'()*+,-./0123456789;<=>?@ABCDEFGHIJKLMNopqrstuvwxyz[_^_');
14617:   Const ('TableXX'('+'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz');
14618:   Function DecodeTriplet( const Value : AnsiString; Delimiter : Char) : AnsiString;
14619:   Function DecodeQuotedPrintable( const Value : AnsiString) : AnsiString;
14620:   Function DecodeURL( const Value : AnsiString) : AnsiString;
14621:   Function EncodeTriplet( const Value : AnsiString; Delimiter : Char; Specials : TSpecials) : AnsiString;
14622:   Function EncodeQuotedPrintable( const Value : AnsiString) : AnsiString;
14623:   Function EncodeSafeQuotedPrintable( const Value : AnsiString) : AnsiString;
14624:   Function EncodeURLElement( const Value : AnsiString) : AnsiString;
14625:   Function EncodeURL( const Value : AnsiString) : AnsiString;
14626:   Function Decode4to3( const Value, Table : AnsiString) : AnsiString;
14627:   Function Decode4to3Ex( const Value, Table : AnsiString) : AnsiString;
14628:   Function Encode3to4( const Value, Table : AnsiString) : AnsiString;
14629:   Function synDecodeBase64( const Value : AnsiString) : AnsiString;
14630:   Function synEncodeBase64( const Value : AnsiString) : AnsiString;
14631:   Function DecodeBase64mod( const Value : AnsiString) : AnsiString;
14632:   Function EncodeBase64mod( const Value : AnsiString) : AnsiString;
14633:   Function DecodeUU( const Value : AnsiString) : AnsiString;
14634:   Function EncodeUU( const Value : AnsiString) : AnsiString;
14635:   Function DecodeXX( const Value : AnsiString) : AnsiString;
14636:   Function DecodeYEnc( const Value : AnsiString) : AnsiString;
14637:   Function UpdateCrc32( Value : Byte; Crc32 : Integer) : Integer;
14638:   Function synCrc32( const Value : AnsiString) : Integer;
14639:   Function UpdateCrc16( Value : Byte; Crc16 : Word) : Word;
14640:   Function Crc16( const Value : AnsiString) : Word;
14641:   Function synMD5( const Value : AnsiString) : AnsiString;
14642:   Function HMAC_MD5( Text, Key : AnsiString) : AnsiString;
14643:   Function MD5LongHash( const Value : AnsiString; Len : integer) : AnsiString;
14644:   Function synSHA1( const Value : AnsiString) : AnsiString;
14645:   Function HMAC_SHA1( Text, Key : AnsiString) : AnsiString;
14646:   Function SHA1LongHash( const Value : AnsiString; Len : integer) : AnsiString;
14647:   Function synMD4( const Value : AnsiString) : AnsiString;
14648: end;
14649:
14650: procedure SIRegister_synachar(CL: TPSPascalCompiler);
14651: begin
14652:   AddTypeS('TTimeChar', '( ISO_8859_1, ISO_8859_2, ISO_8859_3, ISO_8859_4, I'
14653:   + 'SO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8, ISO_8859_9, ISO_8859_10, IS'
14654:   + 'O_8859_13, ISO_8859_14, ISO_8859_15, CP1250, CP1251, CP1252, CP1253, CP125'

```

```

14655:  + '4, CP1255, CP1256, CP1257, CP1258, KOI8_R, CP895, CP852, UCS_2, UCS_4, UTF'
14656:  + '_8, UTF_7, UTF_7mod, UCS_2LE, UCS_4LE, UTF_16, UTF_16LE, UTF_32, UTF_32LE, '
14657:  + 'C99, JAVA, ISO_8859_16, KOI8_U, KOI8_RU, CP862, CP866, MAC, MACCE, MACICE'
14658:  + ', MACCRO, MACRO, MACCYR, MACUK, MACGR, MACTU, MACHEB, MACAR, MACTH, ROMAN8'
14659:  + ', NEXTSTEP, ARMSCII, GEORGIAN_AC, GEORGIAN_PS, KOI8_T, MULELAO, CP1133, T'
14660:  + 'IS620, CP874, VISCII, TCVN, ISO_IR_14, JIS_X0201, JIS_X0208, JIS_X0212, GB'
14661:  + '1988_80, GB2312_80, ISO_IR_165, ISO_IR_149, EUC_JP, SHIFT_JIS, CP932, ISO_'
14662:  + '2022_JP, ISO_2022_JP1, ISO_2022_JP2, GB2312, CP936, GB18030, ISO_2022_CN, '
14663:  + 'ISO_2022_CNE, HZ, EUC_TW, BIG5, CP950, BIG5_HKSCS, EUC_KR, CP949, CP1361, '
14664:  + 'ISO_2022_KR, CP737, CP775, CP853, CP855, CP857, CP858, CP860, CP861, CP863'
14665:  + ', CP864, CP865, CP869, CP1125 )';
14666:  AddTypeS('TTimeSetChar', 'set of TTimeChar');
14667:  Function CharSetConversion(const Value: AnsiString; CharFrom: TTimeChar; CharTo: TTimeChar): AnsiString;
14668:  Function CharSetConversionEx(const Value: AnsiString; CharFrom: TTimeChar; CharTo: TTimeChar; const
TransformTable : array of Word) : AnsiString);
14669:  Function CharSetConversionTrans( Value : AnsiString; CharFrom : TTimeChar; CharTo : TTimeChar; const
TransformTable : array of Word; Translit : Boolean) : AnsiString);
14670:  Function GetCurCP : TTimeChar);
14671:  Function GetCurOEMCP : TTimeChar);
14672:  Function GetCPFromID( Value : AnsiString) : TTimeChar);
14673:  Function GetIDFromCP( Value : TTimeChar) : AnsiString);
14674:  Function NeedCharSetConversion( const Value : AnsiString) : Boolean);
14675:  Function IdealCharSetCoding(const Value: AnsiString; CharFrom: TTimeChar; CharTo: TTimeSetChar): TTimeChar;
14676:  Function GetBOM( Value : TTimeChar) : AnsiString);
14677:  Function StringToWide( const Value : AnsiString) : WideString);
14678:  Function WideToString( const Value : WideString) : AnsiString);
14679: end;
14680:
14681: procedure SIRegister_synamisc(CL: TPSPascalCompiler);
14682: begin
14683:   AddTypeS('TProxySetting', 'record Host : string; Port : string; Bypass : string; end');
14684:   Procedure WakeOnLan( MAC, IP : string);
14685:   Function GetDNS : string);
14686:   Function GetIEProxy( protocol : string) : TProxySetting);
14687:   Function GetLocalIPs : string);
14688: end;
14689:
14690:
14691: procedure SIRegister_synaser(CL: TPSPascalCompiler);
14692: begin
14693:   AddConstantN('synCR', 'Char').SetString( # $0d);
14694:   Const('synLF', 'Char').SetString( # $0a);
14695:   Const('cSerialChunk', 'LongInt').SetInt( 8192);
14696:   Const('LockfileDirectory', 'String').SetString( '/var/lock');
14697:   Const('PortIsClosed', 'LongInt').SetInt( - 1);
14698:   Const('ErrAlreadyOwned', 'LongInt').SetInt( 9991);
14699:   Const('ErrAlreadyInUse', 'LongInt').SetInt( 9992);
14700:   Const('ErrWrongParameter', 'LongInt').SetInt( 9993);
14701:   Const('ErrPortNotOpen', 'LongInt').SetInt( 9994);
14702:   Const('ErrNoDeviceAnswer', 'LongInt').SetInt( 9995);
14703:   Const('ErrMaxBuffer', 'LongInt').SetInt( 9996);
14704:   Const('ErrTimeout', 'LongInt').SetInt( 9997);
14705:   Const('ErrNotRead', 'LongInt').SetInt( 9998);
14706:   Const('ErrFrame', 'LongInt').SetInt( 9999);
14707:   Const('ErrOverrun', 'LongInt').SetInt( 10000);
14708:   Const('ErrRxOver', 'LongInt').SetInt( 10001);
14709:   Const('ErrRxParity', 'LongInt').SetInt( 10002);
14710:   Const('ErrTxFull', 'LongInt').SetInt( 10003);
14711:   Const('dcb_Binary', 'LongWord').SetUInt( $00000001);
14712:   Const('dcb_ParityCheck', 'LongWord').SetUInt( $00000002);
14713:   Const('dcb_OutxCtsFlow', 'LongWord').SetUInt( $00000004);
14714:   Const('dcb_OutxDsrFlow', 'LongWord').SetUInt( $00000008);
14715:   Const('dcb_DtrControlMask', 'LongWord').SetUInt( $00000030);
14716:   Const('dcb_DtrControlDisable', 'LongWord').SetUInt( $00000000);
14717:   Const('dcb_DtrControlEnable', 'LongWord').SetUInt( $00000010);
14718:   Const('dcb_DtrControlHandshake', 'LongWord').SetUInt( $00000020);
14719:   Const('dcb_DsrSensitivity', 'LongWord').SetUInt( $00000040);
14720:   Const('dcb_TXContinueOnXoff', 'LongWord').SetUInt( $00000080);
14721:   Const('dcb_OutX', 'LongWord').SetUInt( $00000100);
14722:   Const('dcb_InX', 'LongWord').SetUInt( $00000200);
14723:   Const('dcb_ErrorChar', 'LongWord').SetUInt( $00000400);
14724:   Const('dcb_NullStrip', 'LongWord').SetUInt( $00000800);
14725:   Const('dcb_RtsControlMask', 'LongWord').SetUInt( $00003000);
14726:   Const('dcb_RtsControlDisable', 'LongWord').SetUInt( $00000000);
14727:   Const('dcb_RtsControlEnable', 'LongWord').SetUInt( $00001000);
14728:   Const('dcb_RtsControlHandshake', 'LongWord').SetUInt( $00002000);
14729:   Const('dcb_RtsControlToggle', 'LongWord').SetUInt( $00003000);
14730:   Const('dcb_AbortOnError', 'LongWord').SetUInt( $00004000);
14731:   Const('dcb_Reserveds', 'LongWord').SetUInt( $FFFF8000);
14732:   Const('synSB1', 'LongInt').SetInt( 0);
14733:   Const('SB1andHalf', 'LongInt').SetInt( 1);
14734:   Const('synSB2', 'LongInt').SetInt( 2);
14735:   Const('synINVALID_HANDLE_VALUE', 'LongInt').SetInt( THandle ( - 1 ));
14736:   Const('CS7fix', 'LongWord').SetUInt( $0000020);
14737:   AddTypeS('synTDCB', 'record DCBlength : DWORD; BaudRate : DWORD; Flags : Long'
14738:   + 'int; wReserved : Word; XonLim : Word; XoffLim : Word; ByteSize : Byte; Par'
14739:   + 'ity : Byte; StopBits : Byte; XonChar : CHAR; XoffChar : CHAR; ErrorChar : '
14740:   + 'CHAR; EofChar : CHAR; EvtChar : CHAR; wReserved1 : Word; end');
14741:   //AddTypeS('PDCB', '^TDCB // will not work');

```

```

14742: //Const('MaxRates','LongInt').SetInt( 18);
14743: //Const('MaxRates','LongInt').SetInt( 30);
14744: //Const('MaxRates','LongInt').SetInt( 19);
14745: Const('O_SYNC','LongWord').SetUInt( $0080);
14746: Const('synOK','LongInt').SetInt( 0);
14747: Const('synErr','LongInt').SetInt( integer ( - 1 ));
14748: AddTypeS('THookSerialReason', '( HR_SerialClose, HR_Connect, HR_CanRead, HR_CanWrite, HR_ReadCount,
HR_WriteCount, HR_Wait )');
14749: Type('THookSerialStatus', Procedure(Sender: TObject; Reason:THookSerialReason; const Value:string));
14750: SIRegister_ESynaSerError(CL);
14751: SIRegister_TBBlockSerial(CL);
14752: Function GetSerialPortNames : string;
14753: end;
14754:
14755: procedure SIRegister_synaicnv(CL: TPSPascalCompiler);
14756: begin
14757: Const('DLLIconvName','String').SetString( 'libiconv.so');
14758: Const('DLLIconvName','String').SetString( 'iconv.dll');
14759: AddTypeS('size_t', 'Cardinal');
14760: AddTypeS('iconv_t', 'Integer');
14761: //AddTypeS('iconv_t', 'Pointer');
14762: AddTypeS('argptr', 'iconv_t');
14763: Function SynaIcnvOpen( const tocode, fromcode : Ansistring) : iconv_t;
14764: Function SynaIcnvOpenTranslit( const tocode, fromcode : Ansistring) : iconv_t;
14765: Function SynaIcnvOpenIgnore( const tocode, fromcode : Ansistring) : iconv_t;
14766: Function SynaIcnv( cd : iconv_t; inbuf : AnsiString; var outbuf : AnsiString) : integer;
14767: Function SynaIcnvClose( var cd : iconv_t) : integer;
14768: Function SynaIcnvCtl( cd : iconv_t; request : integer; argument : argptr) : integer;
14769: Function IsIcnvloaded : Boolean;
14770: Function InitIcnvInterface : Boolean;
14771: Function DestroyIcnvInterface : Boolean;
14772: Const('ICONV_TRIVIALP','LongInt').SetInt( 0);
14773: Const('ICONV_GET_TRANSLITERATE','LongInt').SetInt( 1);
14774: Const('ICONV_SET_TRANSLITERATE','LongInt').SetInt( 2);
14775: Const('ICONV_GET_DISCARD_ILSEQ','LongInt').SetInt( 3);
14776: Const('ICONV_SET_DISCARD_ILSEQ','LongInt').SetInt( 4);
14777: end;
14778:
14779: procedure SIRegister_pingsend(CL: TPSPascalCompiler);
14780: begin
14781: Const('ICMP_ECHO','LongInt').SetInt( 8);
14782: Const('ICMP_ECHOREPLY','LongInt').SetInt( 0);
14783: Const('ICMP_UNREACH','LongInt').SetInt( 3);
14784: Const('ICMP_TIME_EXCEEDED','LongInt').SetInt( 11);
14785: Const('ICMP6_ECHO','LongInt').SetInt( 128);
14786: Const('ICMP6_ECHOREPLY','LongInt').SetInt( 129);
14787: Const('ICMP6_UNREACH','LongInt').SetInt( 1);
14788: Const('ICMP6_TIME_EXCEEDED','LongInt').SetInt( 3);
14789: AddTypeS('TICMPError', '( IE_NoError, IE_Other, IE_TTLExceed, IE_UnreachOt'
+ 'her, IE_UnreachRoute, IE_UnreachAdmin, IE_UnreachAddr, IE_UnreachPort )');
14791: SIRegister_TPINGSend(CL);
14792: Function PingHost( const Host : string) : Integer;
14793: Function TraceRouteHost( const Host : string) : string;
14794: end;
14795:
14796: procedure SIRegister_asnlutil(CL: TPSPascalCompiler);
14797: begin
14798: AddConstantN('synASN1_BOOL','LongWord').SetUInt( $01);
14799: Const('synASN1_INT','LongWord').SetUInt( $02);
14800: Const('synASN1_OCTSTR','LongWord').SetUInt( $04);
14801: Const('synASN1_NULL','LongWord').SetUInt( $05);
14802: Const('synASN1_OBJID','LongWord').SetUInt( $06);
14803: Const('synASN1_ENUM','LongWord').SetUInt( $0a);
14804: Const('synASN1_SEQ','LongWord').SetUInt( $30);
14805: Const('synASN1_SETOF','LongWord').SetUInt( $31);
14806: Const('synASN1_IPADDR','LongWord').SetUInt( $40);
14807: Const('synASN1_COUNTER','LongWord').SetUInt( $41);
14808: Const('synASN1_GAUGE','LongWord').SetUInt( $42);
14809: Const('synASN1_TIMETICKS','LongWord').SetUInt( $43);
14810: Const('synASN1_OPAQUE','LongWord').SetUInt( $44);
14811: Function synASNEncOIDItem( Value : Integer) : AnsiString;
14812: Function synASNDecOIDItem( var Start : Integer; const Buffer : AnsiString) : Integer;
14813: Function synASNEncLen( Len : Integer) : AnsiString;
14814: Function synASNDecLen( var Start : Integer; const Buffer : AnsiString) : Integer;
14815: Function synASNEncInt( Value : Integer) : AnsiString;
14816: Function synASNEncUInt( Value : Integer) : AnsiString;
14817: Function synASNObject( const Data : AnsiString; ASNType : Integer) : AnsiString;
14818: Function synASNItem( var Start:Integer;const Buffer:AnsiString;var ValueType:Integer):AnsiString;
14819: Function synMibToId( Mib : String) : AnsiString;
14820: Function synIdToMib( const Id : AnsiString) : String;
14821: Function synIntMibToStr( const Value : AnsiString) : AnsiString;
14822: Function ASNdump( const Value : AnsiString) : AnsiString;
14823: Function GetMailServers( const DNSHost, Domain : AnsiString; const Servers:TStrings): Boolean;
14824: Function LDAPResultDump( const Value : TLdapResultList) : AnsiString;
14825: end;
14826:
14827: procedure SIRegister_ldapsend(CL: TPSPascalCompiler);
14828: begin
14829: Const('cLDAPProtocol','String').SetString( '389');

```



```

14830:  Const('LDAP_ASN1_BIND_REQUEST','LongWord').SetUInt( $60);
14831:  Const('LDAP_ASN1_BIND_RESPONSE','LongWord').SetUInt( $61);
14832:  Const('LDAP_ASN1_UNBIND_REQUEST','LongWord').SetUInt( $42);
14833:  Const('LDAP_ASN1_SEARCH_REQUEST','LongWord').SetUInt( $63);
14834:  Const('LDAP_ASN1_SEARCH_ENTRY','LongWord').SetUInt( $64);
14835:  Const('LDAP_ASN1_SEARCH_DONE','LongWord').SetUInt( $65);
14836:  Const('LDAP_ASN1_SEARCH_REFERENCE','LongWord').SetUInt( $73);
14837:  Const('LDAP_ASN1_MODIFY_REQUEST','LongWord').SetUInt( $66);
14838:  Const('LDAP_ASN1_MODIFY_RESPONSE','LongWord').SetUInt( $67);
14839:  Const('LDAP_ASN1_ADD_REQUEST','LongWord').SetUInt( $68);
14840:  Const('LDAP_ASN1_ADD_RESPONSE','LongWord').SetUInt( $69);
14841:  Const('LDAP_ASN1_DEL_REQUEST','LongWord').SetUInt( $4A);
14842:  Const('LDAP_ASN1_DEL_RESPONSE','LongWord').SetUInt( $6B);
14843:  Const('LDAP_ASN1_MODIFYDN_REQUEST','LongWord').SetUInt( $6C);
14844:  Const('LDAP_ASN1_MODIFYDN_RESPONSE','LongWord').SetUInt( $6D);
14845:  Const('LDAP_ASN1_COMPARE_REQUEST','LongWord').SetUInt( $6E);
14846:  Const('LDAP_ASN1_COMPARE_RESPONSE','LongWord').SetUInt( $6F);
14847:  Const('LDAP_ASN1_ABANDON_REQUEST','LongWord').SetUInt( $70);
14848:  Const('LDAP_ASN1_EXT_REQUEST','LongWord').SetUInt( $77);
14849:  Const('LDAP_ASN1_EXT_RESPONSE','LongWord').SetUInt( $78);
14850:  SIRegister_TLDAPAttribute(CL);
14851:  SIRegister_TLDAPAttributeList(CL);
14852:  SIRegister_TLDAPResult(CL);
14853:  SIRegister_TLDAPResultList(CL);
14854:  AddTypeS('TLDAPModifyOp', '( MO_Add, MO_Delete, MO_Replace )');
14855:  AddTypeS('TLDAPSearchScope', '( SS_BaseObject, SS_SingleLevel, SS_WholeSubtree )');
14856:  AddTypeS('TLDAPSearchAliases', '( SA_NeverDeref, SA_InSearching, SA_FindingBaseObj, SA_Always )');
14857:  SIRegister_TLDAPSend(CL);
14858:  Function LDAPResultDump( const Value : TLDAPResultList) : AnsiString;
14859:  end;
14860:
14861:
14862: procedure SIRegister_slogsend(CL: TPSPascalCompiler);
14863: begin
14864:   Const('cSysLogProtocol','String').SetString( '514');
14865:   Const('FCL_Kernel','LongInt').SetInt( 0);
14866:   Const('FCL_UserLevel','LongInt').SetInt( 1);
14867:   Const('FCL_MailSystem','LongInt').SetInt( 2);
14868:   Const('FCL_System','LongInt').SetInt( 3);
14869:   Const('FCL_Security','LongInt').SetInt( 4);
14870:   Const('FCL_Syslogd','LongInt').SetInt( 5);
14871:   Const('FCL_Printer','LongInt').SetInt( 6);
14872:   Const('FCL_News','LongInt').SetInt( 7);
14873:   Const('FCL_UUCP','LongInt').SetInt( 8);
14874:   Const('FCL_Clock','LongInt').SetInt( 9);
14875:   Const('FCL_Authorization','LongInt').SetInt( 10);
14876:   Const('FCL_FTP','LongInt').SetInt( 11);
14877:   Const('FCL_NTP','LongInt').SetInt( 12);
14878:   Const('FCL_LogAudit','LongInt').SetInt( 13);
14879:   Const('FCL_LogAlert','LongInt').SetInt( 14);
14880:   Const('FCL_Time','LongInt').SetInt( 15);
14881:   Const('FCL_Local0','LongInt').SetInt( 16);
14882:   Const('FCL_Local1','LongInt').SetInt( 17);
14883:   Const('FCL_Local2','LongInt').SetInt( 18);
14884:   Const('FCL_Local3','LongInt').SetInt( 19);
14885:   Const('FCL_Local4','LongInt').SetInt( 20);
14886:   Const('FCL_Local5','LongInt').SetInt( 21);
14887:   Const('FCL_Local6','LongInt').SetInt( 22);
14888:   Const('FCL_Local7','LongInt').SetInt( 23);
14889:   AddTypeS('TSyslogSeverity', '( Emergency, Alert, Critical, Error, Warning,
14890:   + Notice, Info, Debug )');
14891:   SIRegister_TSyslogMessage(CL);
14892:   SIRegister_TSyslogSend(CL);
14893:   Function ToSysLog(const SyslogServer:string;Facil:Byte;Sever:TSyslogSeverity;const
Content:string):Boolean;
14894:   end;
14895:
14896:
14897: procedure SIRegister_mimemess(CL: TPSPascalCompiler);
14898: begin
14899:   AddTypeS('TMessPriority', '( MP_unknown, MP_low, MP_normal, MP_high )');
14900:   SIRegister_TMessHeader(CL);
14901:   //AddTypeS('TMessHeaderClass', 'class of TMessHeader');
14902:   SIRegister_TMimeMess(CL);
14903: end;
14904:
14905: procedure SIRegister_mimepart(CL: TPSPascalCompiler);
14906: begin
14907:   (FindClass('TOBJECT'),'TMimePart');
14908:   AddTypeS('THookWalkPart', 'Procedure ( const Sender : TMimePart)');
14909:   AddTypeS('TMimePrimary', '( MP_TEXT, MP_MULTIPART, MP_MESSAGE, MP_BINARY )');
14910:   AddTypeS('TMimeEncoding', '( ME_7BIT, ME_8BIT, ME_QUOTED_PRINTABLE, ME_BASE64, ME_UU, ME_XX )');
14911:   SIRegister_TMimePart(CL);
14912:   Const('MaxMimeType','LongInt').SetInt( 25);
14913:   Function GenerateBoundary : string;
14914: end;
14915:
14916: procedure SIRegister_mimeinln(CL: TPSPascalCompiler);
14917: begin

```

```

14918: Function InlineDecode( const Value : string; CP : TMimeChar ) : string'';
14919: Function InlineEncode( const Value : string; CP, MimeP : TMimeChar ) : string'';
14920: Function NeedInline( const Value : AnsiString ) : boolean'';
14921: Function InlineCodeEx( const Value : string; FromCP : TMimeChar ) : string'';
14922: Function InlineCode( const Value : string ) : string'';
14923: Function InlineEmailEx( const Value : string; FromCP : TMimeChar ) : string'';
14924: Function InlineEmail( const Value : string ) : string'';
14925: end;
14926:
14927: procedure SIRegister_ftpsend(CL: TPSPascalCompiler);
14928: begin
14929:   Const('cFtpProtocol','String').SetString( '21');
14930:   Const('cFtpDataProtocol','String').SetString( '20');
14931:   Const('FTP_OK','LongInt').SetInt( 255);
14932:   Const('FTP_ERR','LongInt').SetInt( 254);
14933:   AddTypeS('TFTPStatus', 'Procedure ( Sender : TObject; Response : Boolean; const Value : string)');
14934:   SIRegister_TFTPListRec(CL);
14935:   SIRegister_TFTPList(CL);
14936:   SIRegister_TFTPSend(CL);
14937:   Function FtpGetFile( const IP, Port, FileName, LocalFile, User, Pass : string ) : Boolean'';
14938:   Function FtpPutFile( const IP, Port, FileName, LocalFile, User, Pass : string ) : Boolean'';
14939:   Function FtpInterServerTransfer(const FromIP,FromPort,FromFile,FromUser,FromPass : string; const ToIP,
ToPort,ToFile,ToUser,ToPass : string ) : Boolean'';
14940: end;
14941:
14942: procedure SIRegister_httpsend(CL: TPSPascalCompiler);
14943: begin
14944:   Const('cHttpProtocol','String').SetString( '80');
14945:   AddTypeS('TTransferEncoding', '( TE_UNKNOWN, TE_IDENTITY, TE_CHUNKED )');
14946:   SIRegister_THTTPSend(CL);
14947:   Function HttpGetText( const URL : string; const Response : TStrings ) : Boolean'';
14948:   Function HttpGetBinary( const URL : string; const Response : TStream ) : Boolean'';
14949:   Function HttpPostBinary( const URL : string; const Data : TStream ) : Boolean'';
14950:   Function HttpPostURL( const URL, URLData : string; const Data : TStream ) : Boolean'';
14951:   Function HttpPostFile(const URL,FieldName,FileName:string;const Data:TStream;const
ResultData:TStrings):Boolean;
14952: end;
14953:
14954: procedure SIRegister_smtpsend(CL: TPSPascalCompiler);
14955: begin
14956:   Const('cSmtpProtocol','String').SetString( '25');
14957:   SIRegister_TSMTPSend(CL);
14958:   Function SendToRaw(const MailFrom,MailTo,SMTPHost:string;const MailData:TStrings;const Username,
Password:string) : Bool;
14959:   Function SendTo(const MailFrom,MailTo,Subject,SMTPHost:string;const MailData:TStrings):Boolean;
14960:   Function SendToEx(const MailFrom, MailTo, Subject, SMTPHost : string; const MailData : TStrings; const
Username, Password : string):Boolean'';
14961: end;
14962:
14963: procedure SIRegister_snmpsend(CL: TPSPascalCompiler);
14964: begin
14965:   Const('cSnmpProtocol','String').SetString( '161');
14966:   Const('cSnmpTrapProtocol','String').SetString( '162');
14967:   Const('SNMP_V1','LongInt').SetInt( 0);
14968:   Const('SNMP_V2C','LongInt').SetInt( 1);
14969:   Const('SNMP_V3','LongInt').SetInt( 3);
14970:   Const('PDUGetRequest','LongWord').SetUInt( $A0);
14971:   Const('PDUGetNextRequest','LongWord').SetUInt( $A1);
14972:   Const('PDUGetResponse','LongWord').SetUInt( $A2);
14973:   Const('PDUSetRequest','LongWord').SetUInt( $A3);
14974:   Const('PDUTrap','LongWord').SetUInt( $A4);
14975:   Const('PDUGetBulkRequest','LongWord').SetUInt( $A5);
14976:   Const('PDUInformRequest','LongWord').SetUInt( $A6);
14977:   Const('PDUTrapV2','LongWord').SetUInt( $A7);
14978:   Const('PDUReport','LongWord').SetUInt( $A8);
14979:   Const('ENoError','LongInt').SetInt( 0);
14980:   Const('ETooBig','LongInt').SetInt( 1);
14981:   Const('ENoSuchName','LongInt').SetInt( 2);
14982:   Const('EBadValue','LongInt').SetInt( 3);
14983:   Const('EReadOnly','LongInt').SetInt( 4);
14984:   Const('EGenErr','LongInt').SetInt( 5);
14985:   Const('ENoAccess','LongInt').SetInt( 6);
14986:   Const('EWrongType','LongInt').SetInt( 7);
14987:   Const('EWrongLength','LongInt').SetInt( 8);
14988:   Const('EWrongEncoding','LongInt').SetInt( 9);
14989:   Const('EWrongValue','LongInt').SetInt( 10);
14990:   Const('ENoCreation','LongInt').SetInt( 11);
14991:   Const('EInconsistentValue','LongInt').SetInt( 12);
14992:   Const('EResourceUnavailable','LongInt').SetInt( 13);
14993:   Const('ECommitFailed','LongInt').SetInt( 14);
14994:   Const('EUndoFailed','LongInt').SetInt( 15);
14995:   Const('EAuthorizationError','LongInt').SetInt( 16);
14996:   Const('ENotWritable','LongInt').SetInt( 17);
14997:   Const('EInconsistentName','LongInt').SetInt( 18);
14998:   AddTypeS('TV3Flags', '( NoAuthNoPriv, AuthNoPriv, AuthPriv )');
14999:   AddTypeS('TV3Auth', '( AuthMD5, AuthSHA1 )');
15000:   AddTypeS('TV3Priv', '( PrivDES, Priv3DES, PrivAES )');
15001:   SIRegister_TSNMPMib(CL);
15002:   AddTypeS('TV3Sync', 'record EngineID : AnsiString; EngineBoots : integer; '

```

```

15003:      +'EngineTime : integer; EngineStamp : Cardinal; end');
15004:      SIRegister_TSNMPRec(CL);
15005:      SIRegister_TSNMPSend(CL);
15006:      Function SNMPGet( const OID, Community, SNMPHost : AnsiString; var Value : AnsiString) : Boolean);
15007:      Function SNMPSet( const OID, Community, SNMPHost, Value : AnsiString; ValueType : Integer) : Boolean);
15008:      Function SNMPGetNext(var OID:AnsiString;const Community,SNMPHost:AnsiString;var Value:AnsiString):Boolean;
15009:      Function SNMPGetTable(const BaseOID, Community, SNMPHost : AnsiString; const Value : TStrings): Boolean;
15010:      Function SNMPGetTableElement(const BaseOID,RowID,ColID,Community,SNMPHost:AnsiStr;var Value:AnsiStr):
Boolean;
15011:      Function SendTrap(const Dest,Source, Enterprise, Community : AnsiString; Generic, Specific, Seconds :
Integer; const MIBName, MIBValue : AnsiString; MIBType : Integer) : Integer);
15012:      Function RecvTrap( var Dest, Source, Enterprise, Community : AnsiString; var Generic, Specific, Seconds :
Integer; const MIBName, MIBValue : TStringList) : Integer);
15013: end;
15014:
15015: procedure SIRegister_NetWork(CL: TPSPascalCompiler);
15016: begin
15017:   Function GetDomainName2: AnsiString);
15018:   Function GetDomainController( Domain : AnsiString) : AnsiString);
15019:   Function GetDomainUsers( Controller : AnsiString) : AnsiString);
15020:   Function GetDomainGroups( Controller : AnsiString) : AnsiString);
15021:   Function GetDateTime( Controller : AnsiString) : TDateTime);
15022:   Function GetFullName2( Controller, UserID : AnsiString) : AnsiString);
15023: end;
15024:
15025: procedure SIRegister_wwSystem(CL: TPSPascalCompiler);
15026: begin
15027:   AddTypeS('TwwDateOrder', '( doMDY, doDMY, doYMD )');
15028:   AddTypeS('TwwDateTimeSelection', '( wwdsDay, wwdsMonth, wwdsYear, wwdsHour'
15029:   +', wwdsMinute, wwdsSecond, wwdsAMPM )');
15030:   Function wwStrToDate( const S : string) : boolean);
15031:   Function wwStrToTime( const S : string) : boolean);
15032:   Function wwStrToDateTime( const S : string) : boolean);
15033:   Function wwStrToTimeVal( const S : string) : TDateTime);
15034:   Function wwStrToDateVal( const S : string) : TDateTime);
15035:   Function wwStrToDateTimeVal( const S : string) : TDateTime);
15036:   Function wwStrToInt( const S : string) : boolean);
15037:   Function wwStrToFloat( const S : string) : boolean);
15038:   Function wwGetDateOrder( const DateFormat : string) : TwwDateOrder);
15039:   Function wwNextDay( Year, Month, Day : Word) : integer);
15040:   Function wwPriorDay( Year, Month, Day : Word) : integer);
15041:   Function wwDoEncodeDate( Year, Month, Day : Word; var Date : TDateTime) : Boolean);
15042:   Function wwDoEncodeTime( Hour, Min, Sec, MSec : Word; var Time : TDateTime) : Boolean);
15043:   Function wwGetDateTimeCursorPosition(SelStart:int;Text:string;TimeOnly:Bool):TwwDateTimeSelection);
15044:   Function wwGetTimeCursorPosition( SelStart : integer; Text : string) : TwwDateTimeSelection);
15045:   Function wwScanDate( const S : string; var Date : TDateTime) : Boolean);
15046:   Function wwScanDateEpoch( const S : string; var Date : TDateTime; Epoch : integer) : Boolean);
15047:   Procedure wwSetDateTimeCursorPosition(dateCursor:TwwDateTimeSelection;edit:TCustomEdit;TimeOnly:Bool;
15048:   Function wwStrToFloat2( const S : string; var FloatValue : Extended; DisplayFormat : string) : boolean);
15049: end;
15050:
15051: unit uPSI_Themes;
15052: Function ThemeServices : TThemeServices);
15053: Function ThemeControl( AControl : TControl) : Boolean);
15054: Function UnthemedDesigner( AControl : TControl) : Boolean);
15055: procedure SIRegister_UDDIHelper(CL: TPSPascalCompiler);
15056: begin
15057:   Function GetBindingkeyAccessPoint( const Operator : String; const key : String) : String);
15058: end;
15059: Unit uPSC_menus;
15060: CL.AddDelphiFunction('Function StripHotkey( const Text : string) : string');
15061: Function GetHotkey( const Text : string) : string);
15062: Function AnsiSameCaption( const Text1, Text2 : string) : Boolean);
15063: Function IsAltGRPressed : boolean);
15064:
15065: procedure SIRegister_IdIMAP4Server(CL: TPSPascalCompiler);
15066: begin
15067:   TCommandEvent', 'Procedure(Thread:TidPeerThread; const Tag,CmdStr:String;var Handled:Boolean);
15068:   SIRegister_TidIMAP4Server(CL);
15069: end;
15070:
15071:
15072: Functions_max hex in the box maXbox
15073: functionslist.txt
15074: FunctionsList1 3.9.9.86/88/91
15075:
15076: *****
15077: Procedure
15078: SIZE 7401 6792 6310 5971 4438 3797 3600 3385 3296 2883 2255
15079: Procedure *****Now the Procedure list*****
15080: Procedure ( ACol, ARow : Integer; Items : TStrings)
15081: Procedure ( Agg : TAggregate)
15082: Procedure ( ASender : TComponent; const AReplyStatus : TReplyStatus)
15083: Procedure ( ASender : TComponent; const AString : String; var AMsg : TidMessage)
15084: Procedure ( ASender : TComponent; var AMsg : TidMessage)
15085: Procedure ( ASender : TObject; const ABytes : Integer)
15086: Procedure ( ASender : TObject; VStream : TStream)
15087: Procedure ( AThread : TidThread)
15088: Procedure ( AWebModule : TComponent)

```

```

15089: Procedure ( Column : TColumn)
15090: Procedure ( const AUsername : String; const APassword : String; AAuthenticationResult : Boolean)
15091: Procedure ( const iStart : integer; const sText : string)
15092: Procedure ( Control : TCustomTabControl; TabIndex : Integer; const Rect : TRect; Active : Boolean)
15093: Procedure ( Database : TDatabase; LoginParams : TStrings)
15094: Procedure (DataSet:TCustomClientDataSet;E:EReconcileError;UpdateKind:TUpdateKind;var Action:
TReconcileAction)
15095: Procedure ( DATASET : TDATASET)
15096: Procedure ( DataSet:TDataSet; E:EDatabaseError;UpdateKind:TUpdateKind;var UpdateAction: TUpdateAction)
15097: Procedure ( DATASET : TDATASET; E : TObject; var ACTION : TDATAACTION)
15098: Procedure ( DataSet : TDataSet; UpdateKind : TUpdateKind; var UpdateAction : TUpdateAction)
15099: Procedure ( DATASET : TDATASET; var ACCEPT : BOOLEAN)
15100: Procedure ( DBCtrlGrid : TDBCtrlGrid; Index : Integer)
15101: Procedure ( Done : Integer)
15102: Procedure ( HeaderControl : TCustomHeaderControl; Section : THeaderSection)
15103: Procedure (HeaderControl:TCustomHeaderControl;Section:THeaderSection; const Rect:TRect;Pressed:Boolean)
15104: Procedure
(HeaderControl:TCustomHeaderControl;Section:THeaderSection;Width:Integer;State:TSectionTrackState)
15105: Procedure ( HeaderControl : THeaderControl; Section : THeaderSection)
15106: Procedure (HeaderControl:THeaderControl;Section:THeaderSection; const Rect:TRect; Pressed : Boolean)
15107: Procedure (HeaderControl:THeaderControl;Section:THeaderSection;Width:Integer;State : TSectionTrackState)
15108: Procedure (Sender: TCustomListView;const ARect : TRect;Stage:TCustomDrawStage;var DefaultDraw: Boolean)
15109: Procedure ( Sender : TCustomListView; const ARect : TRect; var DefaultDraw : Boolean)
15110: Procedure ( Sender : TCustomListView; Item : TListItem; Rect : TRect; State : TOwnerDrawState)
15111: Procedure ( Sender : TCustomTreeView; const ARect : TRect; var DefaultDraw : Boolean)
15112: Procedure ( SENDER : TFIELD; const TEXT : String)
15113: Procedure ( SENDER : TFIELD; var TEXT : STRING; DISPLAYTEXT : BOOLEAN)
15114: Procedure ( Sender : TIdTelnet; const Buffer : String)
15115: Procedure ( Sender : TIdTelnet; Status : TIdTelnetCommand)
15116: Procedure ( SENDER : TObject; ACANVAS : TCanvas; ARECT : TRect; SELECTED : BOOLEAN)
15117: Procedure ( SENDER : TObject; ACANVAS : TCanvas; ARECT : TRect; STATE : TOWNERDRAWSTATE)
15118: Procedure ( SENDER : TObject; ACANVAS : TCanvas; var WIDTH, HEIGHT : INTEGER)
15119: Procedure ( Sender : TObject; ACol, ARow : Longint; const Value : string)
15120: Procedure ( Sender : TObject; ACol, ARow : Longint; Rect : TRect; State : TGridDrawState)
15121: Procedure ( Sender : TObject; ACol, ARow : Longint; var CanSelect : Boolean)
15122: Procedure ( Sender : TObject; ACol, ARow : Longint; var Value : string)
15123: Procedure ( Sender : TObject; Button : TMPBtnType)
15124: Procedure ( Sender : TObject; Button : TMPBtnType; var DoDefault : Boolean)
15125: Procedure ( Sender : TObject; Button : TUDBtnType)
15126: Procedure ( Sender : TObject; Canvas : TCanvas; PageRect : TRect; var DoneDrawing : Boolean)
15127: Procedure ( Sender : TObject; ClientSocket : TServerClientWinSocket; var SocketThread : TServerClientThread)
15128: Procedure ( Sender : TObject; Column : TListColumn)
15129: Procedure ( Sender : TObject; Column : TListColumn; Point : TPoint)
15130: Procedure ( Sender : TObject; Connecting : Boolean)
15131: Procedure (Sender:TObject;const PapSize:SmallInt;const Orient:TPrinterOrient;const PageTy:TPageTy;var
DoneDrawing:Bool
15132: Procedure (Sender:TObject; const Rect : TRect; DataCol : Integer; Column: TColumn; State : TGridDrawState)
15133: Procedure ( Sender : TObject; const Rect : TRect; Field : TField; State : TGridDrawState)
15134: Procedure (Sender: TObject; const UserString:string;var DateAndTime:TDateTime;var AllowChange:Boolean)
15135: Procedure ( Sender : TObject; E : Exception; var Handled : Boolean)
15136: Procedure ( Sender : TObject; FromIndex, ToIndex : Longint)
15137: Procedure ( Sender : TObject; FromSection, ToSection : THeaderSection; var AllowDrag : Boolean)
15138: Procedure ( Sender : TObject; Index : LongInt)
15139: Procedure ( Sender : TObject; Item : TListItem)
15140: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange)
15141: Procedure ( Sender : TObject; Item : TListItem; Change : TItemChange; var AllowChange : Boolean)
15142: Procedure ( Sender : TObject; Item : TListItem; Selected : Boolean)
15143: Procedure ( Sender : TObject; Item : TListItem; var AllowEdit : Boolean)
15144: Procedure ( Sender : TObject; Item : TListItem; var S : string)
15145: Procedure ( Sender : TObject; Item1, Item2 : TListItem; Data : Integer; var Compare : Integer)
15146: Procedure ( Sender : TObject; ModalResult : TModalResult; var CanClose : Boolean)
15147: Procedure ( Sender : TObject; Month : LongWord; var MonthBoldInfo : LongWord)
15148: Procedure ( Sender : TObject; NewTab : Integer; var AllowChange : Boolean)
15149: Procedure ( Sender : TObject; Node : TTreeNode)
15150: Procedure ( Sender : TObject; Node : TTreeNode; var AllowChange : Boolean)
15151: Procedure ( Sender : TObject; Node : TTreeNode; var AllowCollapse : Boolean)
15152: Procedure ( Sender : TObject; Node : TTreeNode; var AllowEdit : Boolean)
15153: Procedure ( Sender : TObject; Node : TTreeNode; var AllowExpansion : Boolean)
15154: Procedure ( Sender : TObject; Node : TTreeNode; var S : string)
15155: Procedure ( Sender : TObject; Node1, Node2 : TTreeNode; Data : Integer; var Compare : Integer)
15156: Procedure ( Sender : TObject; NumObjects, NumChars : Integer; var SaveClipboard : Boolean)
15157: Procedure ( Sender : TObject; Rect : TRect)
15158: Procedure ( Sender : TObject; Request : TWebRequest; Response : TWebResponse; var Handled : Boolean)
15159: Procedure (Sender:TObject;Shift:TShiftState;X,Y:Integer;Orient:TPageScrollerOrientation;var Delta:Int
15160: Procedure ( Sender : TObject; Socket : TCustomWinSocket)
15161: Procedure (Sender:TObject; Socket:TCustomWinSocket;ErrorEvent: TErrorEvent; var ErrorCode : Integer)
15162: Procedure ( Sender : TObject; Socket : TCustomWinSocket; SocketEvent : TSocketEvent)
15163: Procedure ( Sender : TObject; Socket : TCustomWinSocket; var ClientSocket : TServerClientWinSocket)
15164: Procedure ( SENDER : TObject; SOURCE : TMENUIITEM; REBUILD : BOOLEAN)
15165: Procedure ( Sender : TObject; StartPos, EndPos : Integer; var AllowChange : Boolean)
15166: Procedure ( Sender : TObject; TabCanvas : TCanvas; R : TRect; Index : Integer; Selected : Boolean)
15167: Procedure ( Sender : TObject; TabIndex : Integer; var ImageIndex : Integer)
15168: Procedure ( Sender : TObject; Thread : TServerClientThread)
15169: Procedure ( Sender : TObject; TickCount : Cardinal; var Reset : Boolean)
15170: Procedure ( Sender : TObject; Username, Password : string)
15171: Procedure ( Sender : TObject; var AllowChange : Boolean)
15172: Procedure ( Sender : TObject; var AllowChange : Boolean; NewValue : SmallInt; Direction:TUpDownDirection)
15173: Procedure ( Sender : TObject; var Caption : string; var Alignment : THTMLCaptionAlignment)
15174: Procedure ( Sender : TObject; var Continue : Boolean)

```



```

15175: Procedure (Sender:TObject;var dest:string;var NumRedirect:Int;var Handled:bool; var VMethod:TidHTTPMethod)
15176: Procedure ( Sender : TObject; var Username : string)
15177: Procedure ( Sender : TObject; Wnd : HWND)
15178: Procedure ( Sender : TToolBar; Button : TToolButton)
15179: Procedure ( Sender : TToolBar; const ARect : TRect; Stage : TCustomDrawStage; var DefaultDraw : Boolean)
15180: Procedure ( Sender : TToolBar; const ARect : TRect; var DefaultDraw : Boolean)
15181: Procedure ( Sender : TToolBar; Index : Integer; var Allow : Boolean)
15182: Procedure ( Sender : TToolBar; Index : Integer; var Button : TToolButton)
15183: Procedure ( StatusBar : TCustomStatusBar; Panel : TStatusPanel; const Rect : TRect)
15184: Procedure ( StatusBar : TStatusPanel; Panel : TStatusPanel; const Rect : TRect)
15185: Procedure (var FieldNames:TWideStrings;SQL:WideString; var BindAllFields:Boolean;var TableName: WideString)
15186: Procedure ( var FieldNames : TWideStrings; SQL : WideString; var TableName : WideString)
15187: procedure (Sender: TObject)
15188: procedure (Sender: TObject; var Done: Boolean)
15189: procedure (Sender: TObject; var Key: Word; Shift: TShiftState);
15190: procedure _T(Name: tbtString; v: Variant);
15191: Procedure AbandonSignalHandler( RtlSigNum : Integer)
15192: Procedure Abort
15193: Procedure About1Click( Sender : TObject)
15194: Procedure Accept( Socket : TSocket)
15195: Procedure AESSymmetricExecute(const plaintext, ciphertext, password: string)
15196: Procedure AESEncryptFile(const plaintext, ciphertext, password: string)
15197: Procedure AESDecryptFile(const replaintext, ciphertext, password: string)
15198: Procedure AESEncryptString(const plaintext: string; var ciphertext: string; password: string)
15199: Procedure AESDecryptString(var plaintext: string; const ciphertext: string; password: string)
15200: Procedure Add( Addend1, Addend2 : TMyBigInt)
15201: Procedure ADD( const AKEY, AVALUE : VARIANT)
15202: Procedure Add( const Key : string; Value : Integer)
15203: Procedure ADD( const NAME, FIELDS : string; OPTIONS : TINDEXTIONS)
15204: Procedure ADD( FIELD : TFIELD)
15205: Procedure ADD( ITEM : TMENUITEM)
15206: Procedure ADD( POPUP : TPOPUPMENU)
15207: Procedure AddCharacters( xCharacters : TCharSet)
15208: Procedure AddDriver( const Name : string; List : TStringList)
15209: Procedure AddImages( Value : TCustomImageList)
15210: Procedure AddIndex( const Name, Fields : string; Options : TIndexOptions; const DescFields : string)
15211: Procedure AddLambdaTransitionTo( oState : TniRegularExpressionState)
15212: Procedure AddLoader( Loader : TBitmapLoader)
15213: Procedure ADDPARAM( VALUE : TPARAM)
15214: Procedure AddPassword( const Password : string)
15215: Procedure AddStandardAlias( const Name, Path, DefaultDriver : string)
15216: Procedure AddState( oState : TniRegularExpressionState)
15217: Procedure AddStrings( Strings : TStringList);
15218: procedure AddStrings(Strings: TStringList);
15219: Procedure AddStrings1( Strings : TWideStrings);
15220: Procedure AddStringTerm( var sString : string; const sTerm : string; const sSeparator : string)
15221: Procedure AddToRecentDocs( const Filename : string)
15222: Procedure AddTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharSet)
15223: Procedure AllFunctionsList1Click( Sender : TObject)
15224: procedure AllObjectsList1Click(Sender: TObject);
15225: Procedure Allocate( AAllocateBytes : Integer)
15226: procedure AllResourceList1Click(Sender: TObject);
15227: Procedure AnsiAppend( var dst : AnsiString; const src : AnsiString)
15228: Procedure AnsiAssign( var dst : AnsiString; var src : AnsiString)
15229: Procedure AnsiDelete( var dst : AnsiString; index, count : Integer)
15230: Procedure AnsiFree( var s : AnsiString)
15231: Procedure AnsiFromWide( var dst : AnsiString; const src : WideString)
15232: Procedure AnsiInsert( var dst : AnsiString; const src : AnsiString; index : Integer)
15233: Procedure AnsiSetLength( var dst : AnsiString; len : Integer)
15234: Procedure AnsiString_to_stream( const Value : ansistring; Destin : TStream)
15235: Procedure AntiFreeze;
15236: Procedure APPEND
15237: Procedure Append( const S : WideString)
15238: procedure Append(S: string);
15239: Procedure AppendByte( var VBytes : TidBytes; AByte : byte)
15240: Procedure AppendBytes( var VBytes : TidBytes; AAdd : TidBytes)
15241: Procedure AppendChunk( Val : OleVariant)
15242: Procedure AppendData( const Data : OleVariant; HitEOF : Boolean)
15243: Procedure AppendStr( var Dest : string; S : string)
15244: Procedure AppendString( var VBytes : TidBytes; const AStr : string; ALength : Integer)
15245: Procedure ApplyRange
15246: procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
15247: Procedure Arrange( Code : TListArrangement)
15248: procedure Assert(expr : Boolean; const msg: string);
15249: procedure Assert2(expr : Boolean; const msg: string);
15250: Procedure Assign( AList : TCustomBucketList)
15251: Procedure Assign( Other : TObject)
15252: Procedure Assign( Source : TDragObject)
15253: Procedure Assign( Source : TPersistent)
15254: Procedure Assign(Source: TPersistent)
15255: procedure Assign2(mystring, mypath: string);
15256: Procedure AssignCurValues( Source : TDataSet);
15257: Procedure AssignCurValues1( const CurValues : Variant);
15258: Procedure ASSIGNFIELD( FIELD : TFIELD)
15259: Procedure ASSIGNFIELDVALUE( FIELD : TFIELD; const VALUE : VARIANT)
15260: Procedure AssignFile(var F: Text; FileName: string)
15261: procedure AssignFile(var F: TextFile; FileName: string)
15262: procedure AssignFileRead(var mystring, myfilename: string);
15263: procedure AssignFileWrite(mystring, myfilename: string);

```

```

15264: Procedure AssignTo( Other : TObject)
15265: Procedure AssignValues( Value : TParameters)
15266: Procedure ASSIGNVALUES( VALUE : TPARAMS)
15267: Procedure AssociateExtension( IconPath, ProgramName, Path, Extension : string)
15268: Procedure Base64_to_stream( const Base64 : ansistring; Destin : TStream)
15269: Procedure Base64ToVar( NatData : Pointer; const SoapData : WideString);
15270: Procedure Base64ToVar1( var V : Variant; const SoapData : WideString);
15271: Procedure BcdAdd( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
15272: Procedure BcdDivide( Dividend, Divisor : string; var bcdOut : TBcd);
15273: Procedure BcdDivide1( const Dividend, Divisor : TBcd; var bcdOut : TBcd);
15274: Procedure BcdDivide2( const Dividend : TBcd; const Divisor : Double; var bcdOut : TBcd);
15275: Procedure BcdDivide3( const Dividend : TBcd; const Divisor : string; var bcdOut : TBcd);
15276: Procedure BcdMultiply( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd);
15277: Procedure BcdMultiply1( const bcdIn : TBcd; const DoubleIn : Double; var bcdOut : TBcd);
15278: Procedure BcdMultiply2( const bcdIn : TBcd; const StringIn : string; var bcdOut : TBcd);
15279: Procedure BcdMultiply3( StringIn1, StringIn2 : string; var bcdOut : TBcd);
15280: Procedure BcdSubtract( const bcdIn1, bcdIn2 : TBcd; var bcdOut : TBcd)
15281: Procedure BcdToBytes( Value : TBcd; Bytes : array of byte)
15282: procedure Beep
15283: Procedure BeepOk
15284: Procedure BeepQuestion
15285: Procedure BeepHand
15286: Procedure BeepExclamation
15287: Procedure BeepAsterisk
15288: Procedure BeepInformation
15289: procedure BEGINDRAG(IMMEDIATE:BOOLEAN)
15290: Procedure BeginLayout
15291: Procedure BeginTimer( const Delay, Resolution : Cardinal)
15292: Procedure BeginUpdate
15293: procedure BeginUpdate;
15294: procedure BigScreen1Click(Sender: TObject);
15295: procedure BinToHex(Buffer: PChar; Text: PChar; BufSize: Integer);
15296: Procedure BitsToBooleans( const Bits : Byte; var B : TBooleanArray; AllBits : Boolean);
15297: Procedure BitsToBooleans1( const Bits : Word; var B : TBooleanArray; AllBits : Boolean);
15298: Procedure BitsToBooleans2( const Bits : Integer; var B : TBooleanArray; AllBits : Boolean);
15299: Procedure BitsToBooleans3( const Bits : Int64; var B : TBooleanArray; AllBits : Boolean);
15300: Procedure BoldDays(Days : array of LongWord; var MonthBoldInfo : LongWord)
15301: Procedure BooleansToBits( var Dest : Byte; const B : TBooleanArray);
15302: Procedure BooleansToBits1( var Dest : Word; const B : TBooleanArray);
15303: Procedure BooleansToBits2( var Dest : Integer; const B : TBooleanArray);
15304: Procedure BooleansToBits3( var Dest : Int64; const B : TBooleanArray);
15305: Procedure BreakPointMenuClick( Sender : TObject)
15306: procedure BRINGTOFRONT
15307: procedure BringToFront;
15308: Procedure btnBackClick( Sender : TObject)
15309: Procedure btnBrowseClick( Sender : TObject)
15310: Procedure BtnClick( Index : TNavigateBtn)
15311: Procedure btnLargeIconsClick( Sender : TObject)
15312: Procedure BuildAndSendRequest( AURI : TIdURI)
15313: Procedure BuildCache
15314: Procedure BurnMemory( var Buff, BuffLen : integer)
15315: Procedure BurnMemoryStream( Destructo : TMemoryStream)
15316: Procedure CalculateFirstSet
15317: Procedure Cancel
15318: procedure CancelDrag;
15319: Procedure CancelEdit
15320: procedure CANCELHINT
15321: Procedure CancelRange
15322: Procedure CancelUpdates
15323: Procedure CancelWriteBuffer
15324: Procedure Capture1(ADest:TStream;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool;
15325: Procedure Capture2( ADest : TStrings; const ADelim : string; const AIsRFCMessage : Boolean);
15326: Procedure Capture3(ADest:TStrings;out VLineCount:Integer;const ADelim:string;const AIsRFCMessage:Bool
15327: procedure CaptureScreenFormat(vname: string; vextension: string);
15328: procedure CaptureScreenPNG(vname: string);
15329: procedure CardinalsToI64(var I: Int64; const LowPart, HighPart: Cardinal);
15330: procedure CASCADE
15331: Procedure CastNativeToSoap(Info:PTypeInfo;var SoapData:WideString;NatData:Pointer;var IsNull: Boolean)
15332: Procedure CastSoapToVariant( SoapInfo : PTypeInfo; const SoapData : WideString; NatData : Pointer);
15333: Procedure cbPathClick( Sender : TObject)
15334: Procedure cbPathKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
15335: Procedure cedebugAfterExecute( Sender : TPSScript)
15336: Procedure cedebugBreakpoint( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
15337: Procedure cedebugCompile( Sender : TPSScript)
15338: Procedure cedebugExecute( Sender : TPSScript)
15339: Procedure cedebugIdle( Sender : TObject)
15340: Procedure cedebugLineInfo( Sender : TObject; const FileName : String; Position, Row, Col : Cardinal)
15341: Procedure CenterHeight( const pc, pcParent : TControl)
15342: Procedure CenterDlg(AForm: TForm; MForm: TForm); { Zentriert Forms }
15343: Procedure CenterForm(AForm: TForm; MForm: TForm); { Zentriert Forms }
15344: Procedure Change
15345: procedure ChangeBiDiModeAlignment(var Alignment: TAlignment);
15346: Procedure Changed
15347: Procedure ChangeDir( const ADirName : string)
15348: Procedure ChangeDirUp
15349: Procedure ChangeEntryTransitions( oNewState : TniRegularExpressionState)
15350: Procedure ChangeLevelBy( Value : TChangeRange)
15351: Procedure ChDir(const s: string)
15352: Procedure Check(Status: Integer)

```

```

15353: Procedure CheckCommonControl( CC : Integer)
15354: Procedure CHECKFIELDNAME( const FIELDNAME : String)
15355: Procedure CHECKFIELDNAMES( const FIELDNAMES : String)
15356: Procedure CheckForDisconnect( const ARaiseExceptionIfDisconnected: boolean; const AIgnoreBuffer: bool)
15357: Procedure CheckForGracefulDisconnect( const ARaiseExceptionIfDisconnected : Boolean)
15358: Procedure CheckToken( T : Char)
15359: procedure CheckToken(t:char)
15360: Procedure CheckTokenSymbol( const S : string)
15361: procedure CheckTokenSymbol(s:string)
15362: Procedure CheckToolMenuDropdown( ToolButton : TToolButton)
15363: procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
15364: Procedure CIED65ToCIED50( var X, Y, Z : Extended)
15365: Procedure CIELABToBGR( const Source, Target : Pointer; const Count : Cardinal);
15366: procedure CipherFileClick(Sender: TObject);
15367: Procedure Clear;
15368: Procedure Clear1Click( Sender : TObject)
15369: Procedure ClearColor( Color : TColor)
15370: Procedure CLEARITEM( AITEM : TMENUITEM)
15371: Procedure ClearMapping
15372: Procedure ClearSelection( KeepPrimary : Boolean)
15373: Procedure ClearWriteBuffer
15374: Procedure Click
15375: Procedure Close
15376: Procedure CloseClick( Sender : TObject)
15377: Procedure CloseDatabase( Database : TDatabase)
15378: Procedure CloseDataSets
15379: Procedure CloseDialog
15380: Procedure CloseFile(var F: Text);
15381: Procedure Closure
15382: Procedure CMYKToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
15383: Procedure CMYKToBGR1( const C, M, Y, K, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
15384: Procedure CodeCompletionList1Click( Sender : TObject)
15385: Procedure ColEnter
15386: Procedure Collapse
15387: Procedure Collapse( Recurse : Boolean)
15388: Procedure ColorRGBToHLS( clrRGB : TColorRef; var Hue, Luminance, Saturation : Word)
15389: Procedure CommaSeparatedToStringList( AList : TStringList; const Value : string)
15390: Procedure CommitFreeAndNil( var Transaction : TDBXTransaction)
15391: Procedure Compile1Click( Sender : TObject)
15392: procedure ComponentCount1Click(Sender: TObject);
15393: Procedure Compress(azipfolder, azipfile: string)
15394: Procedure DeCompress(azipfolder, azipfile: string)
15395: Procedure XZip(azipfolder, azipfile: string)
15396: Procedure XUnZip(azipfolder, azipfile: string)
15397: Procedure Connect( const ATimeout : Integer)
15398: Procedure Connect( Socket : TSocket)
15399: procedure Console1Click(Sender: TObject);
15400: Procedure Continue
15401: Procedure ContinueCount( var Counter : TJclCounter)
15402: procedure CONTROLDESTROYED(CONTROL:TCONTROL)
15403: Procedure ConvertStreamFromAnsiToUTF8( Src, Dst : TStream; cp : integer)
15404: Procedure ConvertStreamFromUTF8ToAnsi( Src, Dst : TStream; cp : integer)
15405: Procedure ConvertImage(vsource, vdestination: string);
15406: // Ex. ConvertImage(Exepath+'my233_bmp.bmp',Exepath+'mypng111.png')
15407: Procedure ConvertToGray(Cnv: TCanvas);
15408: Procedure Copy( Buffer : TRecordBuffer; Dest : TBytes; Offset : Integer; Length : Integer)
15409: Procedure Copy( Buffer : TValueBuffer; Dest : TBytes; Offset : Integer; Count : Integer);
15410: Procedure Copy1( Source : TBytes; Offset : Integer; Buffer : TValueBuffer; Count : Integer);
15411: Procedure CopyBytesToHostLongWord( const ASource: TIdBytes; const ASourceIndex: Integer; var VDest: LongWord)
15412: Procedure CopyBytesToHostWord( const ASource : TIdBytes; const ASourceIndex : Integer; var VDest : Word)
15413: Procedure CopyFrom( mbCopy : TMyBigInt)
15414: Procedure CopyMemoryStream( Source, Destination : TMemoryStream)
15415: procedure CopyRect(const Dest: TRect; Canvas: TCanvas; const Source: TRect);
15416: Procedure CopyTidByteArray( const ASource : array of Byte; const ASourceIndex : Integer; var VDest : array of Byte; const ADestIndex : Integer; const ALength : Integer)
15417: Procedure CopyTidBytes(const ASrc:TIdBytes;const ASrcIndex: Int; var VDest:TIdBytes;const ADestIdx: Int; const ALength: Int)
15418: Procedure CopyTidCardinal( const ASource : Cardinal; var VDest : TIdBytes; const ADestIndex : Integer)
15419: Procedure CopyTidInt64( const ASource : Int64; var VDest : TIdBytes; const ADestIndex : Integer)
15420: Procedure CopyTidIPv6Address(const ASource: TIdIPv6Address; var VDest: TIdBytes; const ADestIndex : Integer)
15421: Procedure CopyTidLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex : Integer)
15422: Procedure CopyTidNetworkLongWord( const ASource : LongWord; var VDest : TIdBytes; const ADestIndex: Integer)
15423: Procedure CopyTidNetworkWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
15424: Procedure CopyTidString(const ASource:String; var VDest:TIdBytes;const ADestIndex:Integer; ALength: Integer)
15425: Procedure CopyTidWord( const ASource : Word; var VDest : TIdBytes; const ADestIndex : Integer)
15426: Procedure CopyToClipboard
15427: Procedure CountParts
15428: Procedure CreateDataSet
15429: Procedure CreateEmptyFile( const FileName : string)
15430: Procedure CreateFileFromString( const FileName, Data : string)
15431: Procedure CreateFromDelta( Source : TPacketDataSet)
15432: procedure CREATEHANDLE
15433: Procedure CreateProcAsUser( const UserDomain, UserName, Password, CommandLine : string)
15434: Procedure CreateProcAsUserEx(const UserDomain,UserName,Password,CommandLine:string;const Environment:PChar)
15435: Procedure CreateTable
15436: Procedure CreateUDLFile( const FileName, ProviderName, DataSourceName : WideString)
15437: procedure CSyntax1Click(Sender: TObject);
15438: Procedure CurrencyToComp( Value : Currency; var Result : Comp)
15439: Procedure CURSORPOSCHANGED

```

```

15440: procedure CutFirstDirectory(var S: String)
15441: Procedure DataBaseError(const Message: string)
15442: Procedure DateTimeToString( var Result : string; Format : string; DateTime : TDateTime);
15443: procedure DateTimeToString(var Result: string; const Format: string; DateTime: TDateTime)
15444: Procedure DateTimeToSystemTime( DateTime : TDateTime; var SystemTime : TSystemTime)
15445: procedure DateTimeToSystemTime(const DateTime: TDateTime; var SystemTime: TSystemTime);
15446: Procedure DBIError(errorCode: Integer)
15447: Procedure DebugOutput( const AText : string)
15448: Procedure DebugRun1Click( Sender : TObject)
15449: procedure Dec;
15450: Procedure DecodeDate( DateTime : TDateTime; var Year, Month, Day : Word)
15451: procedure DecodeDate(const DateTime: TDateTime; var Year, Month, Day: Word);
15452: Procedure DecodeDateDay( const AValue : TDateTime; out AYear, ADayOfYear : Word)
15453: Procedure DecodeDateMonthWeek(const AValue: TDateTime; out AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
15454: Procedure DecodeDateTime(const AValue: TDateTime; out AYear, AMonth, ADay, AHour, AMin, ASec, AMillSec: Word)
15455: Procedure DecodeDateWeek( const AValue : TDateTime; out AYear, AWeekOfYear, ADayOfWeek : Word)
15456: Procedure DecodeDayOfWeekInMonth(const AValue: TDateTime; out AYear, AMonth, ANthDayOfWeek, ADayOfWeek: Word)
15457: Procedure DecodeTime( DateTime : TDateTime; var Hour, Min, Sec, MSec : Word)
15458: procedure DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word);
15459: Procedure Decompile1Click( Sender : TObject)
15460: Procedure DefaultDrawColumnCell(const Rect : TRect; DataCol: Integer; Column: TColumn; State: TGridDrawState)
15461: Procedure DefaultDrawDataCell( const Rect : TRect; Field : TField; State : TGridDrawState)
15462: Procedure DeferLayout
15463: Procedure deffileread
15464: procedure DEFOCUSCONTROL(CONTROL: TWINCONTROL; REMOVING: BOOLEAN)
15465: Procedure DelayMicroseconds( const MicroSeconds : Integer)
15466: Procedure Delete
15467: Procedure Delete( const AFilename : string)
15468: Procedure Delete( const Index : Integer)
15469: Procedure DELETE( INDEX : INTEGER)
15470: Procedure Delete( Index : LongInt)
15471: Procedure Delete( Node : TTreeNode)
15472: procedure Delete(var s: AnyString; ifrom, icount: Longint);
15473: Procedure DeleteAlias( const Name : string)
15474: Procedure DeleteDriver( const Name : string)
15475: Procedure DeleteIndex( const Name : string)
15476: Procedure DeleteKey( const Section, Ident : String)
15477: Procedure DeleteRecords
15478: Procedure DeleteRecords( AffectRecords : TAffectRecords)
15479: Procedure DeleteString( var pStr : String; const pDelStr : string)
15480: Procedure DeleteTable
15481: procedure DelphiSite1Click(Sender: TObject);
15482: Procedure Deselect
15483: Procedure Deselect( Node : TTreeNode)
15484: procedure DestroyComponents
15485: Procedure DestroyHandle
15486: Procedure Diff( var X : array of Double)
15487: procedure Diff(var X: array of Double);
15488: procedure DISABLEALIGN
15489: Procedure DisableConstraints
15490: Procedure Disconnect
15491: Procedure Disconnect( Socket : TSocket)
15492: Procedure Dispose
15493: procedure Dispose(P: PChar)
15494: Procedure DivMod( Dividend : Integer; Divisor : Word; var Result, Remainder : Word)
15495: Procedure DoKey( Key : TDBCtrlGridKey)
15496: Procedure DomToTree(anXmlNode: IXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
15497: Procedure DomToTreeJ(anXmlNode: TJvXMLNode; aTreeNode: TTreeNode; aTreeView: TTreeView);
15498: Procedure Dormant
15499: Procedure DoubleToBcd( const AValue : Double; var bcd : TBcd);
15500: Procedure DoubleToBytes( Value : Double; Bytes : array of byte)
15501: Procedure DoubleToComp( Value : Double; var Result : Comp)
15502: Procedure Draw( Canvas : TCanvas; X, Y, Index : Integer; Enabled : Boolean);
15503: procedure Draw(X, Y: Integer; Graphic: TGraphic);
15504: Procedure Draw1(Canvas: TCanvas; X, Y,
Index: Int; ADrawingStyle: TDrawingStyle; AImageType: TImageType; Enabled: Bool);
15505: Procedure DrawArrow( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
15506: Procedure DrawCheck( ACanvas : TCanvas; Location : TPoint; Size : Integer; Shadow : Boolean)
15507: Procedure DrawChevron( ACanvas : TCanvas; Direction : TScrollDirection; Location : TPoint; Size : Integer)
15508: Procedure DrawColumnCell( const Rect : TRect; DataCol : Integer; Column : TColumn; State : TGridDrawState)
15509: procedure DrawFocusRect(const Rect: TRect);
15510: Procedure DrawHDItoTBitmap( HDIB : THandle; Bitmap : TBitmap)
15511: Procedure DRAWMENUITEM( MENUITEM : TMENUITEM; ACanvas : TCanvas; ARECT : TRECT; STATE : TOWNERDRAWSTATE)
15512: Procedure DrawOverlay(Canvas: TCanvas; X, Y: Integer; ImageIndex: Integer; Overlay: TOverlay; Enabled: Boolean);
15513: Procedure DrawOverlay1(Canvas: TCanvas; X, Y: Int; ImageIndex: Int; Overlay : TOverlay; ADrawingStyle :
TDrawingStyle; AImageType : TImageType; Enabled : Boolean);
15514: procedure drawPlot(vPoints: TPointArray; cFrm: TForm; vcolor: integer);
15515: Procedure DrawPolyLine( const Canvas : TCanvas; var Points : TPointArray; const ClipRect : TRect)
15516: Procedure DropConnections
15517: Procedure DropDown
15518: Procedure DumpDescription( oStrings : TStringList)
15519: Procedure DumpStateTable( oStrings : TStringList)
15520: Procedure EDIT
15521: Procedure EditButtonClick
15522: Procedure EditFont1Click( Sender : TObject)
15523: procedure Ellipse(X1, Y1, X2, Y2: Integer);
15524: Procedure Ellipse1( const Rect : TRect);
15525: Procedure EMMS
15526: Procedure Encode( ADest : TStream)

```



```

15527: procedure ENDDRAG(DROP:BOOLEAN)
15528: Procedure EndEdit( Cancel : Boolean)
15529: Procedure EndTimer
15530: Procedure EndUpdate
15531: Procedure EraseSection( const Section : string)
15532: Procedure Error( const Ident : string)
15533: procedure Error(Ident:Integer)
15534: Procedure ErrorFmt( const Ident : string; const Args : array of const)
15535: Procedure ErrorStr( const Message : string)
15536: procedure ErrorStr(Message:String)
15537: Procedure Exchange( Index1, Index2 : Integer)
15538: procedure Exchange(Index1, Index2: Integer);
15539: Procedure Exec( FileName, Parameters, Directory : string)
15540: Procedure ExecProc
15541: Procedure ExecSQL( UpdateKind : TUpdateKind)
15542: Procedure Execute
15543: Procedure Execute( const CommandText : WideString; var Params, OwnerData : OleVariant)
15544: Procedure ExecuteAndWait( FileName : string; Visibility : Integer)
15545: Procedure ExecuteCommand(executeFile, paramstring: string)
15546: Procedure ExecuteShell(executeFile, paramstring: string)
15547: Procedure ExitThread(ExitCode: Integer); stdcall;
15548: Procedure ExitProcess(ExitCode: Integer); stdcall;
15549: Procedure Expand( AUserName : String; AResults : TStrings)
15550: Procedure Expand( Recurse : Boolean)
15551: Procedure ExportClipboardClick( Sender : TObject)
15552: Procedure ExportDataSetToExcel( DataSet : TDataSet; OnExportProgress : TOnExportProgress)
15553: Procedure ExtractContentFields( Strings : TStrings)
15554: Procedure ExtractCookieFields( Strings : TStrings)
15555: Procedure ExtractFields( Separators, WhiteSpace : TSysCharSet; Content : PChar; Strings : TStrings)
15556: Procedure ExtractHeaderFields(Separ,
    WhiteSpace:TSysChSet;Content:PChar;Strings:TStrings;Decode:Bool;StripQuotes:Bool)
15557: Procedure ExtractHTTFFields(Separators, WhiteSpace:
    TSysCharSet;Content:PChar;Strings:TStrings;StripQuotes:Bool)
15558: Procedure ExtractQueryFields( Strings : TStrings)
15559: Procedure FastDegToGrad
15560: Procedure FastDegToRad
15561: Procedure FastGradToDeg
15562: Procedure FastGradToRad
15563: Procedure FastRadToDeg
15564: Procedure FastRadToGrad
15565: Procedure FileClose( Handle : Integer)
15566: Procedure FileClose(handle: integer)
15567: procedure FilesFromWildcard(Dir,Mask:string;var Files:TStringList;Subdirs,ShowDirs,Multitasking:Bool)
15568: Procedure FileStructure( AStructure : TIdFTPDataStructure)
15569: Procedure FillByte2(var X: Byte ; count: integer; value: byte)
15570: Procedure FillBytes( var VBytes : TIdBytes; const ACount : Integer; const AValue : Byte)
15571: Procedure FillChar( Buffer : TRecordBuffer; Length : Integer; value : Byte)
15572: Procedure FillChar2(var X: PChar ; count: integer; value: char)
15573: Procedure FillCharS(var p: string; count: integer; value: char); //fix3.8
15574: Procedure FillIPList
15575: procedure FillRect(const Rect: TRect);
15576: Procedure FillTStrings( AStrings : TStrings)
15577: Procedure FilterOnBookmarks( Bookmarks : array of const)
15578: procedure FinalizePackage(Module: HMODULE)
15579: procedure FindClose;
15580: procedure FindClose2(var F: TSearchRec)
15581: Procedure FindMatches( const sString : string; xNotify : TniRegularExpressionMatchFoundEvent);
15582: Procedure FindMatches1(const sString:string;iStart:integer;xNotify:TniRegularExpressionMatchFoundEvent);
15583: Procedure FindNearest( const KeyValues : array of const)
15584: Procedure FinishContext
15585: Procedure FIRST
15586: Procedure FloatToDegMinSec( const X : Float; var Degs, Mins, Secs : Float)
15587: Procedure FloatToDecimal(var Result:TFloatRec;const Value:extend;ValueType:TFloatValue;Precis,Decs:Int);
15588: Procedure FloodFill( X, Y : Integer; Color : TColor; FillStyle : TFillStyle)
15589: Procedure FlushSchemaCache( const TableName : string)
15590: procedure FmtStr(var Result: string; const Format: string; const Args: array of const)
15591: procedure FOCUSCONTROL(CONTROL:TWINCONTROL)
15592: Procedure Form1Close( Sender : TObject; var Action : TCloseAction)
15593: Procedure FormActivate( Sender : TObject)
15594: procedure FormatLn(const format: String; const args: array of const); //alias
15595: Procedure FormClose( Sender : TObject; var Action : TCloseAction)
15596: Procedure FormCreate( Sender : TObject)
15597: Procedure FormDestroy( Sender : TObject)
15598: Procedure FormKeyPress( Sender : TObject; var Key : Char)
15599: procedure FormOutput1Click(Sender: TObject);
15600: Procedure FormToHtml( Form : TForm; Path : string)
15601: procedure FrameRect(const Rect: TRect);
15602: Procedure Frame3D(Canvas:TCanvas; var Rect:TRect; TopColor,BottomColor:TColor; Width : Integer)
15603: Procedure NotebookHandlesNeeded( Notebook : TNotebook)
15604: Procedure Free( Buffer : TRecordBuffer)
15605: Procedure Free( Buffer : TValueBuffer)
15606: Procedure Free;
15607: Procedure FreeAndNil(var Obj:TObject)
15608: Procedure FreeImage
15609: procedure FreeMem(P: PChar; Size: Integer)
15610: Procedure FreeTreeData( Tree : TUpdateTree)
15611: Procedure Frexp( const X : Extended; var Mantissa : Extended; var Exponent : Integer)
15612: Procedure FullCollapse
15613: Procedure FullExpand

```

```

15614: Procedure GenerateDPB(sl: TStrings; var DPB: string; var DPBLength: Short); //InterBase
15615: Procedure GenerateTPB(sl: TStrings; var TPB: string; var TPBLength: Short);
15616: Procedure OUTPUTXML( SQLOBJECT : TIBSQL; OUTPUTOBJECT : TIBOUTPUTXML)
15617: Procedure Get1( AURL : string; const AResponseContent : TStream);
15618: Procedure Get1( const ASourceFile : string; ADest : TStream; AResume : Boolean);
15619: Procedure Get2(const ASourceFile,ADestFile: string;const ACanOverwrite: boolean; AResume: Boolean);
15620: Procedure GetAliasNames( List : TStrings)
15621: Procedure GetAliasParams( const AliasName : string; List : TStrings)
15622: Procedure GetApplicationsRunning( Strings : TStrings)
15623: Procedure GetCommandTypes( List : TWideStrings)
15624: Procedure GetConfigParams( const Path, Section : string; List : TStrings)
15625: Procedure GetConnectionNames( List : TStrings; Driver : string; DesignMode : Boolean)
15626: Procedure GetConvFamilies( out AFamilies: TConvFamilyArray)
15627: Procedure GetConvTypes( const AFamily: TConvFamily; out ATypes : TConvTypeArray)
15628: Procedure GetDatabaseNames( List : TStrings)
15629: Procedure GetDataPacket( DataSet : TDataSet; var RecsOut : Integer; out Data : OleVariant)
15630: Procedure GetDIBSizes(Bitmap: HBITMAP; var InfoHeaderSize: longWORD; var ImageSize : longWORD)
15631: Procedure GetDir(d: byte; var s: string)
15632: Procedure GetDirList( const Search : string; List : TStrings; Recursive : Boolean)
15633: Procedure GetDriverNames( List : TStrings)
15634: Procedure GetDriverNames( List : TStrings; DesignMode : Boolean)
15635: Procedure GetDriverParams( const DriverName : string; List : TStrings)
15636: Procedure GetEmails1Click( Sender : TObject)
15637: Procedure getEnvironmentInfo;
15638: Function getEnvironmentString: string;
15639: Procedure GetFieldNames( const DatabaseName, TableName : string; List : TStrings)
15640: Procedure GetFieldNames( const TableName : string; List : TStrings)
15641: Procedure GetFieldNames( const TableName : string; List : TStrings);
15642: Procedure GetFieldNames( const TableName : WideString; List : TWideStrings);
15643: Procedure GETFIELDNAMES( LIST : TSTRINGS)
15644: Procedure GetFieldNames1( const TableName : string; List : TStrings);
15645: Procedure GetFieldNames1( const TableName : string; SchemaName : string; List : TStrings);
15646: Procedure GetFieldNames2( const TableName : WideString; SchemaName : WideString; List : TWideStrings);
15647: Procedure GetFieldNames3( const TableName : WideString; List : TWideStrings);
15648: Procedure GetFileAttributeList( const Items : TStrings; const Attr : Integer)
15649: Procedure GetFileAttributeListEx( const Items : TStrings; const Attr : Integer)
15650: Procedure GetFMTBcd( Buffer : TRecordBuffer; var value : TBcd)
15651: Procedure GetFormatSettings
15652: Procedure GetFromDIB( var DIB : TBitmapInfo)
15653: Procedure GetFromHDIB( HDIB : HBitmap)
15654: Procedure GetIcon( Index : Integer; Image : TIcon);
15655: Procedure GetIcon1(Index : Integer; Image:TIcon; ADrawingStyle:TDrawingStyle; AImageType:TImageType);
15656: Procedure GetIndexInfo( IndexName : string)
15657: Procedure GetIndexNames( const TableName, SchemaName : string; List : TStrings);
15658: Procedure GetIndexNames( List : TStrings)
15659: Procedure GetIndexNames1( const TableName : WideString; List : TWideStrings);
15660: Procedure GetIndexNames2( const TableName, SchemaName : WideString; List : TWideStrings);
15661: Procedure GetIndexNames4( const TableName : string; List : TStrings);
15662: Procedure GetInternalResponse
15663: Procedure GETITEMNAMES( LIST : TSTRINGS)
15664: procedure GetMem(P: PChar; Size: Integer)
15665: Procedure GETOLE2ACCELERATORTABLE(var ACCELTABLE:HACCEL;var ACCELCOUNT:INTEGER;GROUPS:array of INTEGER)
15666: procedure GetPackageDescription(ModuleName: PChar): string)
15667: Procedure GetPackageNames( List : TStrings);
15668: Procedure GetPackageNames1( List : TWideStrings);
15669: Procedure GetParamList( List : TList; const ParamNames : WideString)
15670: Procedure GetProcedureNames( List : TStrings);
15671: Procedure GetProcedureNames( List : TWideStrings);
15672: Procedure GetProcedureNames1( const PackageName : string; List : TStrings);
15673: Procedure GetProcedureNames1( List : TStrings);
15674: Procedure GetProcedureNames2( const PackageName, SchemaName : string; List : TStrings);
15675: Procedure GetProcedureNames3( List : TWideStrings);
15676: Procedure GetProcedureNames4( const PackageName : WideString; List : TWideStrings);
15677: Procedure GetProcedureNames5( const PackageName, SchemaName : WideString; List : TWideStrings);
15678: Procedure GetProcedureParams( ProcedureName : WideString; List : TList);
15679: Procedure GetProcedureParams1( ProcedureName, PackageName : WideString; List : TList);
15680: Procedure GetProcedureParams2( ProcedureName, PackageName, SchemaName : WideString; List : TList);
15681: Procedure GetProviderNames( Names : TWideStrings);
15682: Procedure GetProviderNames( Proc : TGetStrProc)
15683: Procedure GetProviderNames1( Names : TStrings);
15684: procedure GetQRCode2(Width,Height:Word; Correct_Level: string; const Data:string; apath: string);
15685: procedure GetQRCode3(Width,Height:Word; Correct_Level: string; const Data:string; apath: string);//no auto
open image
15686: Function GetQRCode4(Width,Height:Word;Correct_Level:string; const
Data:string;aformat:string):TLinearBitmap;
15687: Procedure GetRGBValue( const Color : TColor; out Red, Green, Blue : Byte)
15688: Procedure GetSchemaNames( List : TStrings);
15689: Procedure GetSchemaNames1( List : TWideStrings);
15690: Procedure GetSessionNames( List : TStrings)
15691: Procedure GetStoredProcNames( const DatabaseName : string; List : TStrings)
15692: Procedure GetStrings( List : TStrings)
15693: Procedure GetSystemTime; stdcall;
15694: Procedure GetTableNames(const DatabaseName,Pattern:string;Extensions,SystemTables:Boolean;List:TStrings)
15695: Procedure GetTableNames( List : TStrings; SystemTables : Boolean)
15696: Procedure GetTableNames( List : TStrings; SystemTables : Boolean);
15697: Procedure GetTableNames( List : TWideStrings; SystemTables : Boolean);
15698: Procedure GetTableNames1( List : TStrings; SchemaName : WideString; SystemTables : Boolean);
15699: Procedure GetTableNames1( List : TStrings; SystemTables : Boolean);
15700: Procedure GetTableNames2( List : TWideStrings; SchemaName : WideString; SystemTables : Boolean);

```

```

15701: Procedure GetTransitionsOn( cChar : char; oStateList : TList)
15702: Procedure GetVisibleWindows( List : Tstrings)
15703: Procedure GoBegin
15704: Procedure GotoCurrent( DataSet : TCustomClientDataSet)
15705: Procedure GotoCurrent( Table : TTable)
15706: procedure GotoEndlClick(Sender: TObject);
15707: Procedure GotoNearest
15708: Procedure GradientFillCanvas(const ACanvas:TCanvas;const AStartCol,AEndCol:TColor;const ARect:TRect;const
Direction: TGradientDirection)
15709: Procedure HandleException( E : Exception; var Handled : Boolean)
15710: procedure HANDLEMESSAGE
15711: procedure HandleNeeded;
15712: Procedure Head( AURL : string)
15713: Procedure Help( var AHelpContents : TStringList; ACommand : string)
15714: Procedure HexToBinary( Stream : TStream)
15715: procedure HexToBinary(Stream:TStream)
15716: Procedure HideDragImage
15717: Procedure HideFormCaption( FormHandle : THandle; Hide : Boolean)
15718: Procedure HideTraybar
15719: Procedure HideWindowForSeconds(secs: integer); { //3 seconds}
15720: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm); { //3 seconds}
15721: Procedure HookOSExceptions
15722: Procedure HookSignal( RtlSigNum : Integer)
15723: Procedure HSLToRGB( const H, S, L : Single; out R, G, B : Single);
15724: Procedure HTMLSyntaxlClick( Sender : TObject)
15725: Procedure IFPS3ClassesPluginlCompImport( Sender : TObject; x : TPSPascalCompiler)
15726: Procedure IFPS3ClassesPluginlExecImport( Sender : TObject; Exec : TPSExec; x : TPSRuntimeClassImporter)
15727: Procedure ImportfromClipboardlClick( Sender : TObject)
15728: Procedure ImportfromClipboard2Click( Sender : TObject)
15729: Procedure IncAMonth( var Year, Month, Day : Word; NumberOfMonths : Integer)
15730: procedure Incb(var x: byte);
15731: Procedure IncludelClick( Sender : TObject)
15732: Procedure IncludeOFF; //preprocessing
15733: Procedure IncludeON;
15734: procedure InfoclClick(Sender: TObject);
15735: Procedure InitAltRecBuffers( CheckModified : Boolean)
15736: Procedure InitContext( Request : TWebRequest; Response : TWebResponse)
15737: Procedure InitContext(WebModuleList:TAbstractWebModuleList;Request:TWebRequest;Response:TWebResponse)
15738: Procedure InitData( ASource : TDataSet)
15739: Procedure InitDelta( ADelta : TPacketDataSet);
15740: Procedure InitDeltal( const ADelta : OleVariant);
15741: Procedure InitErrorPacket( E : EUpdateError; Response : TResolverResponse)
15742: Procedure Initialize
15743: procedure InitializePackage(Module: HMODULE)
15744: Procedure INITIATEACTION
15745: Procedure initHexArray(var hexn: THexArray); //THexArray', 'array[0..15] of char;
15746: Procedure InitKeyFields( Tree : TUpdateTree; ADelta : TPacketDataSet)
15747: Procedure InitModule( AModule : TComponent)
15748: Procedure InitStdConvs
15749: Procedure InitTreeData( Tree : TUpdateTree)
15750: Procedure INSERT
15751: Procedure Insert( Index : Integer; AClass : TClass)
15752: Procedure Insert( Index : Integer; AComponent : TComponent)
15753: Procedure Insert( Index : Integer; AObject : TObject)
15754: Procedure Insert( Index : Integer; const S : WideString)
15755: Procedure Insert( Index : Integer; Image, Mask : TBitmap)
15756: Procedure Insert(Index: Integer; const S: string);
15757: procedure Insert(Index: Integer; S: string);
15758: procedure Insert(s: AnyString; var s2: AnyString; iPos: Longint);
15759: procedure InsertComponent(AComponent:TComponent)
15760: procedure InsertControl(AControl: TControl);
15761: Procedure InsertIcon( Index : Integer; Image : TIcon)
15762: Procedure InsertMasked( Index : Integer; Image : TBitmap; MaskColor : TColor)
15763: Procedure InsertObject( Index : Integer; const S : WideString; AObject : TObject)
15764: procedure InsertObject(Index:Integer;S:String;AObject:TObject)
15765: Procedure Int16ToBytes( Value : SmallInt; Bytes : array of byte)
15766: Procedure Int32ToBytes( Value : Integer; Bytes : array of byte)
15767: Procedure Int64ToBytes( Value : Int64; Bytes : array of byte)
15768: procedure I64ToCardinals(I: Int64; var LowPart, HighPart: Cardinal);
15769: Procedure InternalBeforeResolve( Tree : TUpdateTree)
15770: Procedure InvalidateModuleCache
15771: Procedure InvalidateTitles
15772: Procedure InvalidateDateDayError( const AYear, ADayOfYear : Word)
15773: Procedure InvalidateDateMonthWeekError( const AYear, AMonth, AWeekOfMonth, ADayOfWeek : Word)
15774: Procedure InvalidateDateTimeError(const AYear, AMth, ADay, AHour, AMin, ASec, AMilSec:Word;const
ABaseDate:TDateTime)
15775: Procedure InvalidateDateWeekError(const AYear, AWeekOfYear, ADayOfWeek : Word)
15776: Procedure InvalidateDayOfWeekInMonthError( const AYear, AMonth, ANthDayOfWeek, ADayOfWeek : Word)
15777: procedure JavaSyntaxlClick(Sender: TObject);
15778: Procedure JclLocalesInfoList( const Strings : TStrings; InfoType : Integer)
15779: Procedure KillDataChannel
15780: Procedure LargefontlClick( Sender : TObject)
15781: Procedure LAST
15782: Procedure LaunchCpl( FileName : string)
15783: Procedure Launch( const AFile : string)
15784: Procedure LaunchFile( const AFile : string)
15785: Procedure LetFileList(FileList: TStringlist; apath: string);
15786: Procedure lineToNumber( xmemo : String; met : boolean)
15787: Procedure ListViewCustomDrawItem(Sender:TCustomListView;Item:TListItem;State:TCustomDrawState;var
DefaultDraw:Bool)

```

```

15788: Procedure ListViewCustomDrawSubItem( Sender : TCustomListView; Item : TListItem; SubItem : Integer; State
      : TCustomDrawState; var DefaultDraw : Boolean)
15789: Procedure ListViewData( Sender : TObject; Item : TListItem)
15790: Procedure ListViewDataFind(Sender:TObject; Find : TItemFind; const FindString : String; const FindPosition
      : TPoint; FindData:Pointer; StartIndex:Integer;Direction:TSearchDirection;Wrap:Boolean;var Index: Integer)
15791: Procedure ListViewDataHint( Sender : TObject; StartIndex, EndIndex : Integer)
15792: Procedure ListViewDblClick( Sender : TObject)
15793: Procedure ListViewKeyDown( Sender : TObject; var Key : Word; Shift : TShiftState)
15794: Procedure ListDLLEExports(const FileName: string; List: TStrings);
15795: Procedure Load( const WSDLFileName : WideString; Stream : TMemoryStream)
15796: procedure LoadBytecodeClick(Sender: TObject);
15797: procedure LoadFileFromResource(const FileName: string; ms: TMemoryStream);
15798: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HP)
15799: Procedure LoadFromClipboardFormat( AFormat : Word; AData : THandle; APalette : HPALETTE)
15800: Procedure LoadFromFile( AFileName : string)
15801: Procedure LoadFromFile( const AFileName : string; const AHeadersOnly : Boolean)
15802: Procedure LoadFromFile( const FileName : string)
15803: Procedure LOADFROMFILE( const FILENAME : String; BLOBTYPE : TBLOBTYPE)
15804: Procedure LoadFromFile( const FileName : string; DataType : TDataType)
15805: Procedure LoadFromFile( const FileName : WideString)
15806: Procedure LoadFromFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
15807: Procedure LoadFromFile(const AFileName: string)
15808: procedure LoadFromFile(FileName: string);
15809: procedure LoadFromFile(FileName:String)
15810: Procedure LoadFromResourceID( Instance : THandle; ResID : Integer)
15811: Procedure LoadFromResourceName( Instance : THandle; const ResName : String)
15812: Procedure LoadFromStream( AStream : TStream; const AHeadersOnly : Boolean)
15813: Procedure LoadFromStream( const Stream : TStream)
15814: Procedure LoadFromStream( S : TStream)
15815: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
15816: Procedure LoadFromStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
15817: Procedure LoadFromStream( Stream : TStream)
15818: Procedure LOADFROMSTREAM( STREAM : TSTREAM; BLOBTYPE : TBLOBTYPE)
15819: Procedure LoadFromStream( Stream : TStream; DataType : TDataType)
15820: procedure LoadFromStream(Stream:TStream);
15821: Procedure LoadFromStream1( Stream : TSeekableStream; const FormatExt : string);
15822: Procedure LoadFromStream2( Stream : TStream; const FormatExt : string);
15823: Procedure LoadFromStrings( AStrings : TStrings; const MimeSeparator : Char)
15824: Procedure LoadLastFile1Click( Sender : TObject)
15825: { LoadIcoToImage loads two icons from resource named NameRes,
15826:   into two image lists ALarge and ASmall}
15827: Procedure LoadIcoToImage(ALarge, ASmall: ImgList.TCustomImageList; const NameRes: string);
15828: Procedure LoadMemo
15829: Procedure LoadParamsFromIniFile( FFileName : WideString)
15830: Procedure Lock
15831: Procedure Login
15832: Procedure MakeAlphaChannelFromAlphaPalette( Source : TLinearBitmap)
15833: Procedure MakeAlphaChannelFromColorKey( Source : TLinearBitmap; ColorKey : TColor)
15834: Procedure MakeCaseInsensitive
15835: Procedure MakeDeterministic( var bChanged : boolean)
15836: Procedure MakeGrayPal( var Palette, ColorCount : Integer)
15837: // type TVolumeLevel = 0..127; , saveFilePath as C:\MyFile.wav
15838: Procedure MakeSound(Frequency{Hz}, Duration{mSec}: Int; Volume: TVolumeLevel; savefilePath: string);
15839: Procedure MakeComplexSound(N:integer;freqlist:TStrings;Duration{mSec}:Int;pinknoise:bool;Volume:Byte);
15840: Procedure SetComplexSoundElements(freqedt,Phaseedt,AmpEdt,WaveGrp:integer);
15841: Procedure SetRectComplexFormatStr( const S : string)
15842: Procedure SetPolarComplexFormatStr( const S : string)
15843: Procedure AddComplexSoundObjectToList(newf,newp,newa,news:integer; freqlist: TStrings);
15844: Procedure MakeVisible
15845: Procedure MakeVisible( PartialOK : Boolean)
15846: Procedure Manual1Click( Sender : TObject)
15847: Procedure MarkReachable
15848: Procedure maXbox; //shows the exe version data in a win box
15849: Procedure MeanAndStdDev( const Data : array of Double; var Mean, StdDev : Extended)
15850: Procedure Memo1Change( Sender : TObject)
15851: Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Int;var
      Action:TSynReplaceAction)
15852: Procedure Memo1SpecialLineColors(Sender:TObject;Line:Integer; var Special:Boolean;var FG,BG: TColor)
15853: Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
15854: procedure Memory1Click(Sender: TObject);
15855: Procedure MERGE( MENU : TMAINMENU)
15856: Procedure MergeChangeLog
15857: procedure MINIMIZE
15858: Procedure MinimizeMaxbox;
15859: Procedure MkDir(const s: string)
15860: Procedure mnuPrintFont1Click( Sender : TObject)
15861: procedure ModalStarted
15862: Procedure Modified
15863: Procedure ModifyAlias( Name : string; List : TStrings)
15864: Procedure ModifyDriver( Name : string; List : TStrings)
15865: Procedure MomentsSkewKurtosis(const Data:array of Double;var M1,M2,M3,M4, Skew,Kurtosis:Extended)
15866: Procedure MouseToCell( X, Y : Integer; var ACol, ARow : Longint)
15867: Procedure Move( CurIndex, NewIndex : Integer)
15868: procedure Move(CurIndex, NewIndex: Integer);
15869: procedure Move2(const Source: TByteArray; var Dest: TByteArray; Count: Integer)
15870: Procedure MoveChars( const ASource: String; ASourceStart:int;var ADest: String; ADestStart,ALen:integer)
15871: Procedure moveCube( o : TMyLabel)
15872: Procedure MoveTo( Destination : LongInt; AttachMode : TAttachMode)
15873: procedure MoveTo(X, Y: Integer);

```



```

15874: procedure MoveWindowOrg(DC: HDC; DX, DY: Integer);
15875: Procedure MovePoint(var x,y:Extended; const angle:Extended);
15876: Procedure Multiply( Multiplier1, Multiplier2 : TMyBigInt);
15877: Procedure Multiply1( Multiplier1 : TMyBigInt; Multiplier2 : Integer);
15878: Procedure MsgAbout(Handle: Int; const Msg, Caption: string; const IcoName: string = 'MAINICON'; Flags: DWORD = MB_OK);
15879: Procedure mxButton(x,y,width,height,top,left,ahandle: integer);
15880: Procedure New( Width, Height : Integer; PixFormat : TPixelFormat)
15881: procedure New(P: PChar)
15882: procedure New1Click(Sender: TObject);
15883: procedure NewInstance1Click(Sender: TObject);
15884: Procedure NEXT
15885: Procedure NextMonth
15886: Procedure Noop
15887: Procedure NormalizePath( var APath : string)
15888: procedure ObjectBinaryToText(Input, Output: TStream)
15889: procedure ObjectBinaryToText1(Input, Output: TStream; var OriginalFormat: TStreamOriginalFormat)
15890: procedure ObjectResourceToText(Input, Output: TStream)
15891: procedure ObjectResourceToText1(Input, Output: TStream; var OriginalFormat: TStreamOriginalFormat)
15892: procedure ObjectTextToBinary(Input, Output: TStream)
15893: procedure ObjectTextToBinary1(Input, Output: TStream; var OriginalFormat: TStreamOriginalFormat)
15894: procedure ObjectTextToResource(Input, Output: TStream)
15895: procedure ObjectTextToResource1(Input, Output: TStream; var OriginalFormat: TStreamOriginalFormat)
15896: Procedure Open( const Name, Address, Service : string; Port : Word; Block : Boolean)
15897: Procedure Open( const UserID : WideString; const Password : WideString);
15898: Procedure Open;
15899: Procedure open1Click( Sender : TObject)
15900: Procedure OpenCdDrive
15901: Procedure OpenCloseCdDrive( OpenMode : Boolean; Drive : Char)
15902: Procedure OpenCurrent
15903: Procedure OpenFile(vfilenamepath: string)
15904: Procedure OpenDirectory1Click( Sender : TObject)
15905: Procedure OpenIndexFile( const IndexName : string)
15906: Procedure OpenSchema(const Schema: TSchemaInf; const Restricts: OleVar; const
SchemaID: OleVariant; DataSet: TADODataset)
15907: Procedure OpenWriteBuffer( const AThreshold : Integer)
15908: Procedure OptimizeMem
15909: Procedure Options1( AURL : string);
15910: Procedure OutputDebugString(lpOutputString : PChar)
15911: Procedure PackBuffer
15912: Procedure Paint
15913: Procedure PaintToCanvas( Canvas : TCanvas; const Dest : TRect; HalftoneStretch : Boolean)
15914: Procedure PaintToTBitmap( Target : TBitmap)
15915: Procedure PaletteChanged
15916: Procedure ParentBiDiModeChanged
15917: Procedure PARENTBIDIMODECHANGED( ACONTROL : TOBJECT)
15918: Procedure PasteFromClipboard;
15919: Procedure PasteImage( Source : TLinearBitmap; X, Y : Integer)
15920: Procedure PathExtractElements( const Source : string; var Drive, Path, FileName, Ext : string)
15921: Procedure PerformEraseBackground(Control: TControl; DC: HDC);
15922: Procedure PError( Text : string)
15923: procedure Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
15924: procedure Pie(X1: Integer; Y1: Integer; X2: Integer; Y2: Integer; X3: Int; Y3: Integer; X4: Integer; Y4: Integer);
15925: Procedure Play( FromFrame, ToFrame : Word; Count : Integer)
15926: procedure playmp3(mpath: string);
15927: Procedure PlayMP31Click( Sender : TObject)
15928: Procedure PointCopy( var Dest : TPoint; const Source : TPoint)
15929: Procedure PointMove( var P : TPoint; const DeltaX, DeltaY : Integer)
15930: procedure PolyBezier(const Points: array of TPoint);
15931: procedure PolyBezierTo(const Points: array of TPoint);
15932: procedure Polygon(const Points: array of TPoint);
15933: procedure Polyline(const Points: array of TPoint);
15934: Procedure Pop
15935: Procedure POPULATEOLE2MENU(SHAREDMENU: HMENU; GROUPS: array of INT; var WIDTHS : array of LONGINT)
15936: Procedure PopulationVarianceAndMean( const X : TDynFloatArray; var Variance, Mean : Float)
15937: Procedure POPUP( X, Y : INTEGER)
15938: Procedure PopupURL(URL : WideString);
15939: Procedure POST
15940: Procedure Post4( AURL : string; const ASource : TStrings; const AResponseContent : TStream);
15941: Procedure Post5( AURL : string; const ASource, AResponseContent : TStream);
15942: Procedure Post6( AURL : string; const ASource : TIdMultiPartFormDataStream; AResponseContent : TStream);
15943: Procedure PostUser( const Email, FirstName, LastName : WideString)
15944: Procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
15945: procedure Pred(X: int64);
15946: Procedure Prepare
15947: Procedure PrepareStatement
15948: Procedure PreProcessXML( AList : TStrings)
15949: Procedure PreventDestruction
15950: Procedure Print( const Caption : string)
15951: procedure PrintBitmap(aGraphic: TGraphic; Title: string);
15952: procedure printf(const format: string; const args: array of const);
15953: Procedure PrintList(Value: TStringList);
15954: Procedure PrintImage(aValue: TBitmap; Style: TBitmapStyle); //TBitmapStyle=(bsNormal,bsCentered,bsStretched)
15955: Procedure Printout1Click( Sender : TObject)
15956: Procedure ProcessHeaders
15957: Procedure PROCESSMENUCHAR( var MESSAGE : TWMMENUCHAR)
15958: Procedure ProcessMessage( AMsg : TIdMessage; AHeaderOnly : Boolean);
15959: Procedure ProcessMessage1( AMsg : TIdMessage; const AStream : TStream; AHeaderOnly : Boolean);
15960: Procedure ProcessMessage2( AMsg : TIdMessage; const AFilename : string; AHeaderOnly : Boolean);
15961: Procedure ProcessMessagesOFF; //application.processmessages

```

```

15962: Procedure ProcessMessagesON;
15963: Procedure ProcessPath(const EditText:string;var Drive:Char; var DirPart:string; var FilePart : string)
15964: Procedure ProcessPath1(const EditText:string;var Drive:Char; var DirPart:string; var FilePart:string);
15965: Procedure Proclst Size is: 3797 /1415
15966: Procedure procMessClick( Sender : TObject)
15967: Procedure PSScriptCompile( Sender : TPSScript)
15968: Procedure PSScriptExecute( Sender : TPSScript)
15969: Procedure PSScriptLine( Sender : TObject)
15970: Procedure Push( ABoundary : string)
15971: procedure PushItem(AItem: Pointer)
15972: Procedure Put2( AURL : string; const ASource, AResponseContent : TStream);
15973: Procedure Put2( const ASourceFile : string; const ADestFile : string; const AAppend : boolean);
15974: procedure PutLinuxLines(const Value: string)
15975: Procedure Quit
15976: Procedure RaiseConversionError( const AText : string);
15977: Procedure RaiseConversionError1( const AText : string; const AArgs : array of const);
15978: Procedure RaiseConversionRegError( AFamily : TConvFamily; const ADescription : string)
15979: procedure RaiseException(Ex: TIFException; Param: String);
15980: Procedure RaiseExceptionForLastCmdResult;
15981: procedure RaiseLastException;
15982: procedure RaiseException2;
15983: Procedure RaiseLastOSError
15984: Procedure RaiseLastWin32;
15985: procedure RaiseLastWin32Error)
15986: Procedure RaiseListError( const ATemplate : string; const AData : array of const)
15987: Procedure RandomFillStream( Stream : TMemoryStream)
15988: procedure randomize;
15989: Procedure Rasterize( Rasterizer : TRasterizer; Dst : TBitmap32; DstRect : TRect)
15990: Procedure RCS
15991: Procedure Read( Socket : TSocket)
15992: Procedure ReadBlobData
15993: procedure ReadBuffer(Buffer:string;Count:LongInt)
15994: procedure ReadOnlyClick(Sender: TObject);
15995: Procedure ReadSection( const Section : string; Strings : TStringList)
15996: Procedure ReadSections( Strings : TStringList)
15997: Procedure ReadSections( Strings : TStringList);
15998: Procedure ReadSections1( const Section : string; Strings : TStringList);
15999: Procedure ReadSectionValues( const Section : string; Strings : TStringList)
16000: Procedure ReadStream( AStream : TStream; AByteCount:LongInt; const AReadUntilDisconnect : boolean)
16001: Procedure ReadStrings( ADest : TStringList; AReadLinesCount : Integer)
16002: Procedure ReadVersion2(aFileName: STRING; aVersion : TStringList);
16003: Function ReadVersion(aFileName: STRING; aVersion : TStringList): boolean;
16004: Procedure Realign;
16005: procedure Rectangle(X1, Y1, X2, Y2: Integer);
16006: Procedure Rectangle1( const Rect : TRect);
16007: Procedure RectCopy( var Dest : TRect; const Source : TRect)
16008: Procedure RectFitToScreen( var R : TRect)
16009: Procedure RectGrow( var R : TRect; const Delta : Integer)
16010: Procedure RectGrowX( var R : TRect; const Delta : Integer)
16011: Procedure RectGrowY( var R : TRect; const Delta : Integer)
16012: Procedure RectMove( var R : TRect; const DeltaX, DeltaY : Integer)
16013: Procedure RectMoveTo( var R : TRect; const X, Y : Integer)
16014: Procedure RectNormalize( var R : TRect)
16015: // TFileCallbackProcedure = procedure(filename:string);
16016: Procedure RecurseDirectory(Dir: String;IncludeSubs: boolean;callback: TFileCallbackProcedure);
16017: Procedure RecurseDirectory2(Dir: String; IncludeSubs : boolean);
16018: Procedure RedirectTransition(oOldState:TniRegularExpressionState; oNewState : TniRegularExpressionState)
16019: Procedure Refresh;
16020: Procedure RefreshData( Options : TFetchOptions)
16021: Procedure REFRESHLOOKUPLIST
16022: Procedure RegisterAuthenticationMethod( MethodName : string; AuthClass : TIdAuthenticationClass)
16023: Procedure RegisterChanges( Value : TChangeLink)
16024: Procedure RegisterConversionFormat( const AExtension : string; AConversionClass : TConversionClass)
16025: Procedure RegisterFileFormat( const AExtension, ADescription : string; AGraphicClass : TGraphicClass)
16026: Procedure RegisterFileFormat(Extension,AppID:string;Description:string;Executable:string;IconIndex:Int)
16027: Procedure ReInitialize( ADelay : Cardinal)
16028: procedure RELEASE
16029: Procedure Remove( const AByteCount : integer)
16030: Procedure REMOVE( FIELD : TFIELD)
16031: Procedure REMOVE( ITEM : TMENUITEM)
16032: Procedure REMOVE( POPUP : TPOPUPMENU)
16033: Procedure RemoveAllPasswords
16034: procedure RemoveComponent(AComponent:TComponent)
16035: Procedure RemoveDir( const ADirName : string)
16036: Procedure RemoveLambdaTransitions( var bChanged : boolean)
16037: Procedure REMOVEPARAM( VALUE : TPARAM)
16038: Procedure RemoveTransitionTo( oState : TniRegularExpressionState; xCharacters : TCharSet);
16039: Procedure RemoveTransitionTo1( oState : TniRegularExpressionState);
16040: Procedure Rename( const ASourceFile, ADestFile : string)
16041: Procedure Rename( const FileName : string; Reload : Boolean)
16042: Procedure RenameTable( const NewTableName : string)
16043: Procedure Replace( Index : Integer; Image, Mask : TBitmap)
16044: Procedure ReplaceClick( Sender : TObject)
16045: Procedure ReplaceDate( var DateTime : TDateTime; NewDate : TDateTime)
16046: procedure ReplaceDate(var DateTime: TDateTime; const NewDate: TDateTime))
16047: Procedure ReplaceIcon( Index : Integer; Image : TIcon)
16048: Procedure ReplaceMasked( Index : Integer; NewImage : TBitmap; MaskColor : TColor)
16049: Procedure ReplaceTime( var DateTime : TDateTime; NewTime : TDateTime)
16050: procedure ReplaceTime(var DateTime: TDateTime; const NewTime: TDateTime);

```

```

16051: Procedure Requery( Options : TExecuteOptions)
16052: Procedure Reset
16053: Procedure Reset1Click( Sender : TObject)
16054: Procedure ResizeCanvas( XSiz, YSiz, XPos, YPos : Integer; Color : TColor)
16055: Procedure ResourceExplore1Click(Sender: TObject);
16056: Procedure RestoreContents
16057: Procedure RestoreDefaults
16058: Procedure RestoreOtherInstance( MainFormClassName, MainFormCaption : string)
16059: Procedure RetrieveHeaders
16060: Procedure RevertRecord
16061: Procedure RGBAToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal)
16062: Procedure RGBToBGR( const Source, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
16063: Procedure RGBToBGR1( const R, G, B, Target : Pointer; const BitsPerSample : Byte; Count : Cardinal);
16064: Procedure RGBToHSL( const R, G, B : Single; out H, S, L : Single);
16065: Procedure RGBToHSL1( const RGB : TColor32; out H, S, L : Single);
16066: Procedure RGBToHSV( r, g, b : Integer; var h, s, v : Integer)
16067: Procedure RleCompress2( Stream : TStream)
16068: Procedure RleDecompress2( Stream : TStream)
16069: Procedure Rmdir(const S: string)
16070: Procedure Rollback
16071: Procedure Rollback( TransDesc : TTransactionDesc)
16072: Procedure RollbackFreeAndNil( var Transaction : TDBXTransaction)
16073: Procedure RollbackIncompleteFreeAndNil( var Transaction : TDBXTransaction)
16074: Procedure RollbackTrans
16075: procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer);
16076: Procedure RoundToAllocGranularity64( var Value : Int64; Up : Boolean)
16077: Procedure RoundToAllocGranularityPtr( var Value : Pointer; Up : Boolean)
16078: Procedure RunDll32Internal( Wnd : HWND; const DLLName, FuncName, CmdLine : string; CmdShow : Integer)
16079: Procedure S_AddMessageToStrings( AMessages : TStrings; AMsg : string)
16080: Procedure S_EBox( const AText : string)
16081: Procedure S_GetEncryptionKeys(DateTime1: TDateTime; var StartKey: int; var MultKey: integer; var
AddKey: int
16082: Procedure S_IBox( const AText : string)
16083: Procedure S_ReplaceChar( var cStr : string; cOldChr, cNewChr : char)
16084: Procedure S_ReplaceStringInFile( AFileName : string; ASearchString, AReplaceString : string)
16085: Procedure S-TokenInit( cBuffer : PChar; const cDelimiters : string)
16086: Procedure SampleVarianceAndMean
16087: ( const X : TDynFloatArray; var Variance, Mean : Float)
16088: Procedure Save2Click( Sender : TObject)
16089: Procedure Saveas3Click( Sender : TObject)
16090: Procedure Savebefore1Click( Sender : TObject)
16091: Procedure SaveBytesToFile(const Data: TBytes; const FileName: string);
16092: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
16093: Procedure SaveConfigFile
16094: Procedure SaveOutput1Click( Sender : TObject)
16095: procedure SaveScreenshotClick(Sender: TObject);
16096: Procedure SaveLn(pathname, content: string); //SaveLn(exepath+'mysavelntest.txt', memo2.text);
16097: Procedure SaveToClipboardFormat( var AFormat : Word; var AData : THandle; var APalette : HPALETTE)
16098: Procedure SaveToClipboardFormat( var Format : Word; var Data : THandle; var APalette : HPALETTE)
16099: Procedure SaveToFile( AFileName : string)
16100: Procedure SAVETOFILE( const FILENAME : String)
16101: Procedure SaveToFile( const FileName : WideString)
16102: Procedure SaveToFile( const FileName : WideString; Format : TPersistFormat)
16103: Procedure SaveToFile( const FileName, FileType : string; Bitmap : TLinearBitmap)
16104: procedure SaveToFile(FileName: string);
16105: procedure SaveToFile(FileName:String)
16106: Procedure SaveToStream( AStream : TStream; const AHeadersOnly : Boolean)
16107: Procedure SaveToStream( OutStream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
16108: Procedure SaveToStream( S : TStream)
16109: Procedure SaveToStream( Stream : TSeekableStream; const Ext : string; Bitmap : TLinearBitmap)
16110: Procedure SaveToStream( Stream : TStream)
16111: Procedure SaveToStream( Stream : TStream; Format : TDataPacketFormat)
16112: procedure SaveToStream(Stream: TStream);
16113: procedure SaveToStream(Stream:TStream)
16114: Procedure SaveToStream1( Stream : TSeekableStream; const FormatExt : string);
16115: Procedure SaveToStream2( Stream : TStream; const FormatExt : string);
16116: Procedure SaveToStrings( AStrings : TStrings; const MimeSeparator : Char)
16117: procedure Say(const sText: string)
16118: Procedure SBytecode1Click( Sender : TObject)
16119: Procedure ScaleImage( const SourceBitmap, ResizedBitmap : TBitmap; const ScaleAmount : Double)
16120: procedure ScriptExplorer1Click(Sender: TObject);
16121: Procedure Scroll( Distance : Integer)
16122: Procedure Scroll( DX, DY : Integer)
16123: procedure ScrollBy(DeltaX, DeltaY: Integer);
16124: procedure SCROLLINVIEW(ACONTROL:TCONTROL)
16125: Procedure ScrollTabs( Delta : Integer)
16126: Procedure Search1Click( Sender : TObject)
16127: procedure SearchAndOpenDoc(vfilenamepath: string)
16128: procedure SearchAndOpenFile(vfilenamepath: string)
16129: procedure SearchAndReplace(aStrList: TStrings; aSearchStr, aNewStr: string)
16130: procedure SearchAndCopy(aStrList: TStrings; aSearchStr, aNewStr: string; offset: integer);
16131: Procedure SearchNext1Click( Sender : TObject)
16132: Procedure Select( Node : TTreeNode; ShiftState : TShiftState);
16133: Procedure Select1( const Nodes : array of TTreeNode);
16134: Procedure Select2( Nodes : TList);
16135: Procedure SelectNext( Direction : Boolean)
16136: Procedure SelectNextPage( GoForward : Boolean; CheckTabVisible : Boolean)
16137: Procedure SelfTestPEM //unit uPSI_cPEM
16138: Procedure Send( AMsg : TIdMessage)

```

```

16139: //config first in const MAILINIFILE = 'maildef.ini';
16140: //ex.: SendEmail('max@kleiner.ch','max@kleiner.com','this test7','maxbox the SSL fox',
16141: Procedure SendEmail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
16142: Procedure SendMail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
16143: Procedure SendMsg( AMsg : TIdMessage; const AHeadersOnly : Boolean)
16144: Procedure SendMsg(AMsg: TIdMessage; const AHeadersOnly: Boolean = False)
16145: Procedure SendResponse
16146: Procedure SendStream( AStream : TStream)
16147: Procedure Set8087CW( NewCW : Word)
16148: Procedure SetAll( One, Two, Three, Four : Byte)
16149: Procedure SetAltRecBuffers( Old, New, Cur : PChar)
16150: Procedure SetAppDispatcher( const ADispatcher : TComponent)
16151: procedure SetArrayLength;
16152: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer); //2 dimension
16153: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
16154: Procedure SetAsHandle( Format : Word; Value : THandle)
16155: procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer)
16156: procedure SetCaptureControl(Control: TControl);
16157: Procedure SetColumnAttributes
16158: Procedure SetCookieField(Values:TStrings;const ADomain,APath:string;AExpires:TDateTime;ASecure:Boolean)
16159: Procedure SetCustomHeader( const Name, Value : string)
16160: Procedure SetExprParams(const Text:Widestring;Opts:TFilterOpts;ParserOpts:TParserOpts;const
    FieldName:Widestring)
16161: Procedure SetFMTBcd( Buffer : TRecordBuffer; value : TBcd)
16162: Procedure SetFocus
16163: procedure SetFocus; virtual;
16164: Procedure SetInitialState
16165: Procedure SetKey
16166: procedure SetLastError(ErrorCode: Integer)
16167: procedure SetLength;
16168: Procedure SetLineBreakStyle( var T : Text; Style : TTextLineBreakStyle)
16169: Procedure SETOLE2MENUHANDLE( HANDLE : HMENU)
16170: Procedure SetParams( ADataset : TDataset; UpdateKind : TUpdateKind);
16171: procedure SETPARAMS(APOSITION,AMIN,AMAX:INTEGER)
16172: Procedure SetParams1( UpdateKind : TUpdateKind);
16173: Procedure SetPassword( const Password : string)
16174: Procedure SetPointer( Ptr : Pointer; Size : Longint)
16175: Procedure SetPrimalityTest( const Method : TPrimalityTestMethod)
16176: Procedure SetPrinter( ADevice, ADriver, APort : PChar; ADeviceMode : THandle)
16177: Procedure SetProvider( Provider : TComponent)
16178: Procedure SetProxy( const Proxy : string)
16179: Procedure SetPSResult( var PSResult : TPSResult; Value : TObject)
16180: Procedure SetRange( const StartValues, EndValues : array of const)
16181: Procedure SetRangeEnd
16182: Procedure SetRate( const aPercent, aYear : integer)
16183: procedure SetRate(const aPercent, aYear: integer)
16184: Procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
16185: Procedure SetSafeCallExceptionMsg( Msg : String)
16186: procedure SETSELTEXTBUF(BUFFER:PCHAR)
16187: Procedure SetSize( AWidth, AHeight : Integer)
16188: procedure SetSize(NewSize:LongInt)
16189: procedure SetString(var s: string; buffer: PChar; len: Integer)
16190: Procedure SetStrings( List : TStrings)
16191: Procedure SetText( Text : PwideChar)
16192: procedure SetText(Text: PChar);
16193: Procedure SetTextBuf( Buffer : PChar)
16194: procedure SETTEXTBUF(BUFFER:PCHAR)
16195: Procedure SetTick( Value : Integer)
16196: Procedure SetTimeout( ATimeOut : Integer)
16197: Procedure SetTraceEvent( Event : TDBXTraceEvent)
16198: Procedure SetUserName( const UserName : string)
16199: Procedure SetWallpaper( Path : string);
16200: procedure ShellStyle1Click(Sender: TObject);
16201: Procedure SHORTCUTTOKEY( SHORTCUT : TSHORTCUT; var KEY : WORD; var SHIFT : TSHIFTSTATE)
16202: Procedure ShowFileProperties( const FileName : string)
16203: Procedure ShowInclude1Click( Sender : TObject)
16204: Procedure ShowInterfaces1Click( Sender : TObject)
16205: Procedure ShowLastException1Click( Sender : TObject)
16206: Procedure ShowMessage( const Msg : string)
16207: Procedure ShowMessageBig(const aText : string);
16208: Procedure ShowMessageBig2(const aText : string; autosize: boolean);
16209: Procedure ShowMessageBig3(const aText : string; fsize: byte; autosize: boolean);
16210: Procedure MsgBig(const aText : string; //alias
16211: procedure showMessage(mytext: string);
16212: Procedure ShowMessageFmt( const Msg : string; Params : array of const)
16213: procedure ShowMessageFmt(const Msg: string; Params: array of const))
16214: Procedure ShowMessagePos( const Msg : string; X, Y : Integer)
16215: procedure ShowMessagePos(const Msg: string; X: Integer; Y: Integer))
16216: Procedure ShowSearchDialog( const Directory : string)
16217: Procedure ShowSpecChars1Click( Sender : TObject)
16218: Procedure ShowBitmap(bmap: TBitmap); //draw in a form!
16219: Procedure ShredFile( const FileName : string; Times : Integer)
16220: procedure Shuffle(vQ: TStringList);
16221: Procedure ShuffleList( var List : array of Integer; Count : Integer)
16222: Procedure SimulateKeystroke( Key : byte; Shift : TShiftState)
16223: Procedure SinCos( const Theta : Extended; var Sin, Cos : Extended)
16224: Procedure SinCosE( X : Extended; out Sin, Cos : Extended)
16225: Procedure Site( const ACommand : string)
16226: Procedure SkipEOL

```



```

16227: Procedure Sleep( ATime : cardinal)
16228: Procedure Sleep( milliseconds : Cardinal)
16229: Function SleepEx( dwMilliseconds : DWORD; bAlertable : BOOL) : DWORD
16230: Procedure Slinenumber1Click( Sender : TObject)
16231: Procedure Sort
16232: Procedure SortColorArray(ColorArray:TColorArray;L,R:Int;SortType:TColorArraySortType;Reverse:Bool)
16233: procedure Speak(const sText: string) //async like voice
16234: procedure Speak2(const sText: string) //sync
16235: procedure Split(Str: string; SubStr: string; List: TStringList);
16236: Procedure SplitNameValue( const Line : string; var Name, Value : string)
16237: Procedure SplitColumns( const AData : String; AStrings : TStringList; const ADelim : String)
16238: Procedure SplitColumnsNoTrim( const AData : String; AStrings : TStringList; const ADelim : String)
16239: Procedure SplitLines( AData : PChar; ADataSize : Integer; AStrings : TStringList)
16240: Procedure SplitString( const AStr, AToken : String; var VLeft, VRight : String)
16241: procedure SQLSyntax1Click(Sender: TObject);
16242: Procedure SRand( Seed : RNG_IntType)
16243: Procedure Start
16244: Procedure StartCount( var Counter : TJclCounter; const Compensate : Boolean)
16245: Procedure StartFileFinder3(spath,aext,searchstr: string; arecursiv: boolean; reslist: TStringList);
16246: Procedure StartTransaction( TransDesc : TTransactionDesc)
16247: Procedure Status( var AStatusList : TStringList)
16248: Procedure StatusBar1Db1Click( Sender : TObject)
16249: Procedure StepInto1Click( Sender : TObject)
16250: Procedure StepIt
16251: Procedure StepOut1Click( Sender : TObject)
16252: Procedure Stop
16253: procedure stopmp3;
16254: procedure StartWeb(aurl: string);
16255: Procedure Str(aInt: integer; astr: string); //of system
16256: Procedure StrDispose( Str : PChar)
16257: procedure StrDispose(Str: PChar)
16258: Procedure StrReplace(var Str: String; Old, New: String);
16259: Procedure StretchDIBits( DC : THandle; const Dest : TRect; HalftoneStretch : Boolean)
16260: procedure StretchDraw(const Rect: TRect; Graphic: TGraphic);
16261: Procedure StringToBytes( Value : String; Bytes : array of byte)
16262: procedure StrSet(c : Char; I : Integer; var s : String);
16263: Procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStringList);
16264: Procedure StructureMount( APath : String)
16265: procedure STYLECHANGED(SENDER:TObJEct)
16266: Procedure Subselect( Node : TTreeNode; Validate : Boolean)
16267: procedure Succ(X: int64);
16268: Procedure SumsAndSquares( const Data : array of Double; var Sum, SumOfSquares : Extended)
16269: procedure SwapChar(var X,Y: char); //swapX follows
16270: Procedure SwapFloats( var X, Y : Float)
16271: procedure SwapGrid(grd: TStringGrid);
16272: Procedure SwapOrd( var I, J : Byte);
16273: Procedure SwapOrd( var X, Y : Integer)
16274: Procedure SwapOrd1( var I, J : Shortint);
16275: Procedure SwapOrd2( var I, J : Smallint);
16276: Procedure SwapOrd3( var I, J : Word);
16277: Procedure SwapOrd4( var I, J : Integer);
16278: Procedure SwapOrd5( var I, J : Cardinal);
16279: Procedure SwapOrd6( var I, J : Int64);
16280: Procedure SymetricCompareFiles(const plaintext, replaintext: string)
16281: Procedure Synchronizel( Method : TMethod);
16282: procedure SyntaxCheck1Click(Sender: TObject);
16283: Procedure SysFreeString(const S: WideString); stdcall;
16284: Procedure TakeOver( Other : TLinearBitmap)
16285: Procedure Talkln(const sText: string) //async voice
16286: procedure tbtn6resClick(Sender: TObject);
16287: Procedure tbtnUseCaseClick( Sender : TObject)
16288: procedure TerminalStyle1Click(Sender: TObject);
16289: Procedure Terminate
16290: Procedure texSyntax1Click( Sender : TObject)
16291: procedure TextOut(X, Y: Integer; Text: string);
16292: Procedure TextRect( Rect : TRect; X, Y : Integer; const Text : string);
16293: procedure TextRect(Rect: TRect; X: Integer; Y: Integer; const Text: string);
16294: Procedure TextRect1( var Rect : TRect; var Text : string; TextFormat : TTextFormat);
16295: Procedure TextStart
16296: procedure TILE
16297: Procedure TimeStampToBytes( Value : TBcd; Bytes : array of byte)
16298: Procedure TitleClick( Column : TColumn)
16299: Procedure ToDo
16300: procedure toolbtnTutorialClick(Sender: TObject);
16301: Procedure Trace1( AURL : string; const AResponseContent : TStream);
16302: Procedure TransferMode( ATransferMode : TIdFTPTransferMode)
16303: Procedure Truncate
16304: procedure Tutorial101Click(Sender: TObject);
16305: procedure Tutorial10Statistics1Click(Sender: TObject);
16306: procedure Tutorial11Forms1Click(Sender: TObject);
16307: procedure Tutorial12SQL1Click(Sender: TObject);
16308: Procedure tutorial1Click( Sender : TObject)
16309: Procedure tutorial21Click( Sender : TObject)
16310: Procedure tutorial31Click( Sender : TObject)
16311: Procedure tutorial4Click( Sender : TObject)
16312: Procedure Tutorial5Click( Sender : TObject)
16313: procedure Tutorial6Click(Sender: TObject);
16314: procedure Tutorial91Click(Sender: TObject);
16315: Procedure UnhookSignal( RtlSigNum : Integer; OnlyIfHooked : Boolean)

```

```

16316: procedure UniqueString(var str: AnsiString)
16317: procedure UnloadLoadPackage(Module: HMODULE)
16318: Procedure Unlock
16319: Procedure UNMERGE( MENU : TMAINMENU)
16320: Procedure UnRegisterChanges( Value : TChangeLink)
16321: Procedure UnregisterConversionFamily( const AFamily : TConvFamily)
16322: Procedure UnregisterConversionType( const AType : TConvType)
16323: Procedure UnRegisterProvider( Prov : TCustomProvider)
16324: Procedure UPDATE
16325: Procedure UpdateBatch( AffectRecords : TAffectRecords)
16326: Procedure UPDATECURSORPOS
16327: Procedure UpdateFile
16328: Procedure UpdateItems( FirstIndex, LastIndex : Integer)
16329: Procedure UpdateResponse( AResponse : TWebResponse)
16330: Procedure UpdateScrollBar
16331: Procedure UpdateView1Click( Sender : TObject)
16332: procedure Val(const s: string; var n, z: Integer)
16333: procedure VarArraySet(c : Variant; I : Integer; var s : Variant);
16334: Procedure VarFMTBcdCreate( var ADest : Variant; const ABcd : TBcd);
16335: Procedure VariantAdd( const src : Variant; var dst : Variant)
16336: Procedure VariantAnd( const src : Variant; var dst : Variant)
16337: Procedure VariantArrayRedim( var V : Variant; High : Integer)
16338: Procedure VariantCast( const src : Variant; var dst : Variant; vt : Integer)
16339: Procedure VariantClear( var V : Variant)
16340: Procedure VariantCpy( const src : Variant; var dst : Variant)
16341: Procedure VariantDiv( const src : Variant; var dst : Variant)
16342: Procedure VariantMod( const src : Variant; var dst : Variant)
16343: Procedure VariantMul( const src : Variant; var dst : Variant)
16344: Procedure VariantOr( const src : Variant; var dst : Variant)
16345: Procedure VariantPutElement( var V : Variant; const data : Variant; i1 : integer);
16346: Procedure VariantPutElement1( var V : Variant; const data : Variant; i1, i2 : integer);
16347: Procedure VariantPutElement2( var V : Variant; const data : Variant; i1, i2, i3 : integer);
16348: Procedure VariantPutElement3( var V : Variant; const data : Variant; i1, i2, i3, i4 : integer);
16349: Procedure VariantPutElement4( var V : Variant; const data : Variant; i1, i2, i3, i4, i5 : integer);
16350: Procedure VariantShl( const src : Variant; var dst : Variant)
16351: Procedure VariantShr( const src : Variant; var dst : Variant)
16352: Procedure VariantSub( const src : Variant; var dst : Variant)
16353: Procedure VariantXor( const src : Variant; var dst : Variant)
16354: Procedure VarCastError;
16355: Procedure VarCastError1( const ASourceType, ADestType : TVarType);
16356: Procedure VarInvalidOp
16357: Procedure VarInvalidNullOp
16358: Procedure VarOverflowError( const ASourceType, ADestType : TVarType)
16359: Procedure VarRangeCheckError( const ASourceType, ADestType : TVarType)
16360: Procedure VarArrayCreateError
16361: Procedure VarResultCheck( AResult : HRESULT);
16362: Procedure VarResultCheck1( AResult : HRESULT; ASourceType, ADestType : TVarType);
16363: Procedure HandleConversionException( const ASourceType, ADestType : TVarType)
16364: Function VarTypeAsText( const AType : TVarType) : string
16365: procedure Voice(const sText: string) //async
16366: procedure Voice2(const sText: string) //sync
16367: Procedure WaitMiliseconds( AMSec : word)
16368: Procedure WideAppend( var dst : WideString; const src : WideString)
16369: Procedure WideAssign( var dst : WideString; var src : WideString)
16370: Procedure WideDelete( var dst : WideString; index, count : Integer)
16371: Procedure WideFree( var s : WideString)
16372: Procedure WideFromAnsi( var dst : WideString; const src : AnsiString)
16373: Procedure WideFromPChar( var dst : WideString; src : PChar)
16374: Procedure WideInsert( var dst : WideString; const src : WideString; index : Integer)
16375: Procedure WideSetLength( var dst : WideString; len : Integer)
16376: Procedure WideString2Stream( aWideString : WideString; oStream : TStream)
16377: Procedure WideStringToBytes( Value : WideString; Bytes : array of byte)
16378: Procedure WinColorToOpenGLColor( const Color : TColor; out Red, Green, Blue : Float)
16379: Procedure WinInet_HttpGet(const Url: string; Stream:TStream);
16380: Procedure HttpGet(const Url: string; Stream:TStream);
16381: Procedure WordToTwoBytes( AWord : Word; ByteArray : TIdBytes; Index : integer)
16382: Procedure WordWrap1Click( Sender : TObject)
16383: Procedure Write( const AOut : string)
16384: Procedure Write( Socket : TSocket)
16385: procedure Write(S: string);
16386: Procedure WriteBinaryStream( const Section, Name : string; Value : TStream)
16387: Procedure WriteBool( const Section, Ident : string; Value : Boolean)
16388: Procedure WriteBuffer( const ABuffer, AByteCount : Longint; const AWriteNow : Boolean)
16389: procedure WriteBuffer(Buffer:String;Count:LongInt)
16390: Procedure WriteCardinal( AValue : Cardinal; const AConvert : Boolean)
16391: Procedure WriteChar( AValue : Char)
16392: Procedure WriteDate( const Section, Name : string; Value : TDateTime)
16393: Procedure WriteDateTime( const Section, Name : string; Value : TDateTime)
16394: Procedure WriteFloat( const Section, Name : string; Value : Double)
16395: Procedure WriteHeader( AHeader : TStrings)
16396: Procedure WriteInteger( AValue : Integer; const AConvert : Boolean)
16397: Procedure WriteInteger( const Section, Ident : string; Value : Longint)
16398: Procedure WriteLn( const AOut : string)
16399: procedure Writeln(s: string);
16400: Procedure WriteLog( const FileName, LogLine : string)
16401: Procedure WriteRFCReply( AReply : TIdRFCReply)
16402: Procedure WriteRFCStrings( AStrings : TStrings)
16403: Procedure WriteSmallInt( AValue : SmallInt; const AConvert : Boolean)
16404: Procedure WriteStream(AStream:TStream;const AAll:Bool;const AWriteByteCount:Bool;const ASize:Int)

```

```

16405: Procedure WriteString( const Section, Ident, Value : String)
16406: Procedure WriteStrings( AValue : TStrings; const AWriteLinesCount : Boolean)
16407: Procedure WriteTime( const Section, Name : string; Value : TDateTime)
16408: Procedure WriteObjectResourceHeader( ObjStream, Output : TStream)
16409: Procedure Write16bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
16410: Procedure Write32bitResourceHeader( const AName : TBytes; DataSize : Integer; Output : TStream)
16411: procedure WStrSet(c : AnyString; I : Integer; var s : AnyString);
16412: procedure XMLSyntax1Click(Sender: TObject);
16413: Procedure XOR_Streams2( Dest, Srce : TMemoryStream)
16414: Procedure XOR_Streams3( Dest, SrceA, SrceB : TMemoryStream)
16415: Procedure ZeroFillStream( Stream : TMemoryStream)
16416: procedure XMLSyntax1Click(Sender: TObject);
16417: Procedure ZeroMemory( Ptr : Pointer; Length : Longint)
16418: procedure(Control: TWinControl; Index: Integer; Rect: TRect; State: Byte)
16419: procedure(Control: TWinControl; Index: Integer; var Height: Integer)
16420: procedure(Sender, Source: TObject; X, Y: Integer; State: TDragState; var Accept: Boolean)
16421: procedure(Sender, Source: TObject; X, Y: Integer)
16422: procedure(Sender, Target: TObject; X, Y: Integer)
16423: procedure(Sender: TObject; ASection, AWidth: Integer)
16424: procedure(Sender: TObject; ScrollCode: TScrollCode; var ScrollPos: Integer)
16425: procedure(Sender: TObject; Shift: TShiftState; X, Y: Integer);
16426: procedure(Sender: TObject; var Action: TCloseAction)
16427: procedure(Sender: TObject; var CanClose: Boolean)
16428: procedure(Sender: TObject; var Key: Char);
16429: ProcedureName ProcedureNames ProcedureParametersCursor @
16430:
16431: *****Now Constructors constructor *****
16432: Size is: 1115 996 628 550 544 501 459 (381) (360) (270 246) (235)
16433: Attach( VersionInfoData : Pointer; Size : Integer)
16434: constructor Create( ABuckets : TBucketListSizes)
16435: Create( ACallbackWnd : HWND)
16436: Create( AClient : TCustomTaskDialog)
16437: Create( AClient : TIdTelnet)
16438: Create( ACollection : TCollection)
16439: Create( ACollection : TFavoriteLinkItems)
16440: Create( ACollection : TTaskDialogButtons)
16441: Create( AConnection : TIdCustomHTTP)
16442: Create( ACreateSuspended : Boolean)
16443: Create( ADataSet : TCustomSQLDataSet)
16444: CREATE( ADATASET : TDATASET)
16445: Create( Aggregates : TAggregates; ADataSet : TCustomClientDataSet);
16446: Create( AGrid : TCustomDBGrid)
16447: Create( AGrid : TStringGrid; AIndex : Longint)
16448: Create( AHTTP : TIdCustomHTTP)
16449: Create( AListItems : TListItems)
16450: Create( AOnBytesRemoved : TIdBufferBytesRemoved)
16451: Create( AOnBytesRemoved : TIdBufferBytesRemoved)
16452: Create( AOwner : TCommonCalendar)
16453: Create( AOwner : TComponent)
16454: CREATE( AOWNER : TCOMPONENT)
16455: Create( AOwner : TCustomListView)
16456: Create( AOwner : TCustomOutline)
16457: Create( AOwner : TCustomRichEdit)
16458: Create( AOwner : TCustomRichEdit; AttributeType : TAttributeType)
16459: Create( AOwner : TCustomTreeView)
16460: Create( AOwner : TIdUserManager)
16461: Create( AOwner : TListItems)
16462: Create( AOwner :
TObject;Handle:hDBCurl;CBType:CBType;CBBuf:Pointer;CBBufSize:Int;CallbackEvent:TBDECallbackEvent; Chain :
Boolean)
16463: CREATE( AOWNER : TPERSISTENT)
16464: Create( AOwner : TPersistent)
16465: Create( AOwner : TTable)
16466: Create( AOwner : TTreeNode)
16467: Create( AOwner : TWinControl; const ClassName : string)
16468: Create( AParent : TIdCustomHTTP)
16469: Create( AParent : TUpdateTree; AResolver : TCustomResolver)
16470: Create( AProvider : TBaseProvider)
16471: Create( AProvider : TCustomProvider);
16472: Create( AProvider : TDataSetProvider)
16473: Create( ASocket : TCustomWinSocket; TimeOut : Longint)
16474: Create( ASocket : TSocket)
16475: Create( AStrings : TWideStrings)
16476: Create( AToolBar : TToolBar)
16477: Create( ATreeNodes : TTreeNodes)
16478: Create( Autofill : boolean)
16479: Create( AWebPageInfo : TAbstractWebPageInfo)
16480: Create( AWebRequest : TWebRequest)
16481: Create( Collection : TCollection)
16482: Create( Collection : TIdMessageParts; ABody : TStrings)
16483: Create( Collection : TIdMessageParts; const AFileName : TFileName)
16484: Create( Column : TColumn)
16485: Create( const AConvFamily : TConvFamily; const ADescription : string)
16486: Create( const AConvFamily : TConvFamily; const ADescription : string; const AFactor : Double)
16487: Create( const AConvFamily : TConvFamily; const ADescription : string; const AToCommonProc,
AFromCommonProc : TConversionProc)
16488: Create( const AInitialState : Boolean; const AManualReset : Boolean)
16489: Create( const ATabSet : TTabSet)
16490: Create( const Compensate : Boolean)

```

```

16491: Create( const FileMap : TJclCustomFileMapping; Access, Size : Cardinal; ViewOffset : Int64)
16492: Create( const FileName : string)
16493: Create( const FileName : string; FileMode:Cardinal; const Name:string; Protect:Cardinal; const MaximumSize
: Int64; const SecAttr : PSecurityAttributes);
16494: Create( const FileName : string; FileMode : WordfmsShareDenyWrite)
16495: Create( const MaskValue : string)
16496: Create(const Name:string; Protect:Cardinal;const MaximumSize:Int64;const SecAttr:PSecurityAttributes)
16497: Create( const Prefix : string)
16498: Create( const sRegularExpression : string; xFlags : TniRegularExpressionMatchFlags)
16499: Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
16500: Create( const sRule : string; xFlags : TniRegularExpressionMatchFlags)
16501: Create( CoolBar : TCoolBar)
16502: Create( CreateSuspended : Boolean; ASocket : TServerClientWinSocket)
16503: Create( CreateSuspended : Boolean; ASocket : TServerWinSocket)
16504: Create( DataSet : TDataSet; const
Text:Widestring;Options:TFilterOptions;ParserOptions:TParserOptions;const FieldName : Widestring;
DepFields : TBits; FieldMap : TFieldMap)
16505: Create( DBCtrlGrid : TDBCtrlGrid)
16506: Create( DSTableProducer : TDSTableProducer)
16507: Create( DSTableProducer : TDSTableProducer; ColumnClass : THTMLTableColumnClass)
16508: Create( ErrorCode : DBIResult)
16509: Create( Field : TBlobField; Mode : TBlobStreamMode)
16510: Create( Grid : TCustomDBGrid; ColumnClass : TColumnClass)
16511: Create( HeaderControl : TCustomHeaderControl)
16512: Create( HTTPRequest : TWebRequest)
16513: Create( iStart : integer; sText : string)
16514: Create( iValue : Integer)
16515: Create( Kind : TmTimerKind; Notification : TmNotificationKind)
16516: Create( MciErrNo : MCIERROR; const Msg : string)
16517: Create(MemoryStream:TCustomMemStream;FreeStream:Boolean;const AIndexOption:TJclMappedTextReaderIndex);
16518: Create( Message : string; ErrorCode : DBResult)
16519: Create( Msg : string)
16520: Create( NativeError, Context : string; ErrCode, PrevError : Integer; E : Exception)
16521: Create( NativeError, Context : string; ErrorCode, PreviousError : DBResult)
16522: Create( oExpression : TniRegularExpression; eType : TniRegularExpressionStateType)
16523: Create( oOwner : TniRegularExpression; xFlags : TniRegularExpressionMatchFlags)
16524: Create(oSource:TniRegularExpressState;oDestination:TniRegularExprState;xCharacts:TCharSet;bLambda:bool)
16525: Create( Owner : EDBEngineError; ErrorCode : DBIResult; NativeError : Longint; Message : PChar)
16526: Create( Owner : TCustomComboBoxEx)
16527: CREATE( OWNER : TINDEXTDEFS; const NAME, FIELDS: String; OPTIONS: TINDEXTOPTIONS)
16528: Create( Owner : TPersistent)
16529: Create( Params : TStrings)
16530: Create( Size : Cardinal)
16531: Create( Socket : TSocket; ServerWinSocket : TServerWinSocket)
16532: Create( StatusBar : TCustomStatusBar)
16533: Create( WebDispatcher : TCustomWebDispatcher; ItemClass : TCollectionItemClass)
16534: Create( WebResponse : TWebResponse; ItemClass : TCollectionItemClass)
16535: Create(AHandle:Integer)
16536: Create(AOwner: TComponent); virtual;
16537: Create(const AURI : string)
16538: Create(FileName:String;Mode:Word)
16539: Create(Instance:THandle;ResName:String;ResType:PChar)
16540: Create(Stream : TStream)
16541: Createl( ADataset : TDataset);
16542: Createl(const FileHandle:THandle;const Name:string;Protect:Cardinal;const MaximumSize:Int64;const
SecAttr:PSecurityAttributes);
16543: Createl( const FileName : string; const AIndexOption : TJclMappedTextReaderIndex);
16544: Create2( Other : TObject);
16545: CreateAt(FileMap: TJclCustomFileMapping; Access,Size:Cardinal;ViewOffset:Int64;Address: Pointer)
16546: CreateError(const anErrCode: Integer;const asReplyMessage: string; const asErrorMessage: string)
16547: CreateFmt( MciErrNo : MCIERROR; const Msg : string; const Args : array of const)
16548: CreateFromId(Instance:THandle;ResId:Integer;ResType:PChar)
16549: CreateLinked( DBCtrlGrid : TDBCtrlGrid)
16550: CREATENEW(AOWNER:TCOMPONENT; Dummy: Integer)
16551: CreateRes( Ident : Integer);
16552: CreateRes( MciErrNo : MCIERROR; Ident : Integer)
16553: CreateRes( ResStringRec : PResStringRec);
16554: CreateResHelp( Ident : Integer; AHelpContext : Integer);
16555: CreateResHelp( ResStringRec : PResStringRec; AHelpContext : Integer);
16556: CreateShadow( AOwner : TComponent; ControlSide : TControlSide)
16557: CreateSize( AWidth, AHeight : Integer)
16558: Open( const Name : string; const InheritHandle : Boolean; const DesiredAccess : Cardinal)
16559:
16560: -----
16561: unit uPSI_MathMax;
16562: -----
16563: CONSTS
16564: Bernstein: Float = 0.2801694990238691330364364912307; // Bernstein constant
16565: Cbrt2: Float = 1.2599210498948731647672106072782; // CubeRoot(2)
16566: Cbrt3: Float = 1.4422495703074083823216383107801; // CubeRoot(3)
16567: Cbrt10: Float = 2.1544346900318837217592935665194; // CubeRoot(10)
16568: Cbrt100: Float = 4.6415888336127788924100763509194; // CubeRoot(100)
16569: CbrtPi: Float = 1.4645918875615232630201425272638; // CubeRoot(Pi)
16570: Catalan: Float = 0.9159655941772190150546035149324; // Catalan constant
16571: Pi: Float = 3.1415926535897932384626433832795; // PI
16572: PI: Extended = 3.1415926535897932384626433832795;
16573: PiOn2: Float = 1.5707963267948966192313216916398; // PI / 2
16574: PiOn3: Float = 1.0471975511965977461542144610932; // PI / 3
16575: PiOn4: Float = 0.78539816339744830961566084581988; // PI / 4

```



```

16576: Sqrt2: Float      = 1.4142135623730950488016887242097; // Sqrt(2)
16577: Sqrt3: Float      = 1.7320508075688772935274463415059; // Sqrt(3)
16578: Sqrt5: Float      = 2.2360679774997896964091736687313; // Sqrt(5)
16579: Sqrt10: Float     = 3.1622776601683793319988935444327; // Sqrt(10)
16580: SqrtPi: Float     = 1.7724538509055160272981674833411; // Sqrt(Pi)
16581: Sqrt2Pi: Float    = 2.506628274631000502415765284811; // Sqrt(2 * PI)
16582: TwoPi: Float      = 6.283185307179586476925286766559; // 2 * PI
16583: ThreePi: Float    = 9.4247779607693797153879301498385; // 3 * PI
16584: Ln2: Float        = 0.69314718055994530941723212145818; // Ln(2)
16585: Ln10: Float       = 2.3025850929940456840179914546844; // Ln(10)
16586: LnPi: Float       = 1.1447298858494001741434273513531; // Ln(Pi)
16587: Log2J: Float      = 0.30102999566398119521373889472449; // Log10(2)
16588: Log3: Float       = 0.47712125471966243729502790325512; // Log10(3)
16589: LogPi: Float      = 0.4971498726941338543512682882909; // Log10(Pi)
16590: LogE: Float       = 0.43429448190325182765112891891661; // Log10(E)
16591: E: Float          = 2.7182818284590452353602874713527; // Natural constant
16592: hLn2Pi: Float     = 0.91893853320467274178032973640562; // Ln(2*PI)/2
16593: inv2Pi: Float     = 0.15915494309189533576888376337251436203445964574046; // 0.5/Pi
16594: TwoToPower63: Float = 9223372036854775808.0; // 2^63
16595: GoldenMean: Float = 1.618033988749894848204586834365638; // GoldenMean
16596: EulerMascheroni: Float = 0.5772156649015328606065120900824; // Euler GAMMA
16597: RadCor : Double   = 57.29577951308232; {number of degrees in a radian}
16598: StDelta : Extended = 0.00001; {delta for difference equations}
16599: StEpsilon : Extended = 0.00001; {epsilon for difference equations}
16600: StMaxIterations : Integer = 100; {max attempts for convergence}
16601:
16602: MaxAngle', ( 9223372036854775808.0);
16603: MaxTanH', ( 5678.2617031470719747459655389854);
16604: MaxFactorial', 'LongInt').SetInt( 1754);
16605: MaxFloatingPoint', (1.189731495357231765085759326628E+4932);
16606: MinFloatingPoint', (3.3621031431120935062626778173218E-4932);
16607: MaxTanH', ( 354.89135644669199842162284618659);
16608: MaxFactorial', 'LongInt').SetInt( 170);
16609: MaxFloatingPointD', (1.797693134862315907729305190789E+308);
16610: MinFloatingPointD', (2.2250738585072013830902327173324E-308);
16611: MaxTanH', ( 44.361419555836499802702855773323);
16612: MaxFactorial', 'LongInt').SetInt( 33);
16613: MaxFloatingPointS', ( 3.4028236692093846346337460743177E+38);
16614: MinFloatingPointS', ( 1.1754943508222875079687365372222E-38);
16615: PiExt', ( 3.1415926535897932384626433832795);
16616: RatioDegToRad', ( PiExt / 180.0);
16617: RatioGradToRad', ( PiExt / 200.0);
16618: RatioDegToGrad', ( 200.0 / 180.0);
16619: RatioGradToDeg', ( 180.0 / 200.0);
16620: Crc16PolynomCCITT', 'LongWord').SetUInt( $1021);
16621: Crc16PolynomIBM', 'LongWord').SetUInt( $8005);
16622: Crc16Bits', 'LongInt').SetInt( 16);
16623: Crc16Bytes', 'LongInt').SetInt( 2);
16624: Crc16HighBit', 'LongWord').SetUInt( $8000);
16625: NotCrc16HighBit', 'LongWord').SetUInt( $7FFF);
16626: Crc32PolynomIEEE', 'LongWord').SetUInt( $04C11DB7);
16627: Crc32PolynomCastagnoli', 'LongWord').SetUInt( $1EDC6F41);
16628: Crc32Koopman', 'LongWord').SetUInt( $741B8CD7);
16629: Crc32Bits', 'LongInt').SetInt( 32);
16630: Crc32Bytes', 'LongInt').SetInt( 4);
16631: Crc32HighBit', 'LongWord').SetUInt( $80000000);
16632: NotCrc32HighBit', 'LongWord').SetUInt( $7FFFFFFF);
16633:
16634: MinByte = Low(Byte);
16635: MaxByte = High(Byte);
16636: MinWord = Low(Word);
16637: MaxWord = High(Word);
16638: MinShortInt = Low(ShortInt);
16639: MaxShortInt = High(ShortInt);
16640: MinSmallInt = Low(SmallInt);
16641: MaxSmallInt = High(SmallInt);
16642: MinLongWord = LongWord(Low(LongWord));
16643: MaxLongWord = LongWord(High(LongWord));
16644: MinLongInt = LongInt(Low(LongInt));
16645: MaxLongInt = LongInt(High(LongInt));
16646: MinInt64 = Int64(Low(Int64));
16647: MaxInt64 = Int64(High(Int64));
16648: MinInteger = Integer(Low(Integer));
16649: MaxInteger = Integer(High(Integer));
16650: MinCardinal = Cardinal(Low(Cardinal));
16651: MaxCardinal = Cardinal(High(Cardinal));
16652: MinNativeUInt = NativeUInt(Low(NativeUInt));
16653: MaxNativeUInt = NativeUInt(High(NativeUInt));
16654: MinNativeInt = NativeInt(Low(NativeInt));
16655: MaxNativeInt = NativeInt(High(NativeInt));
16656: Function CosH( const Z : Float) : Float;
16657: Function SinH( const Z : Float) : Float;
16658: Function TanH( const Z : Float) : Float;
16659:
16660:
16661: //*****from DMath.Dll Lib of types.inc in source\dmath_dll
16662: InvLn2 = 1.44269504088896340736; { 1/Ln(2) }
16663: InvLn10 = 0.43429448190325182765; { 1/Ln(10) }
16664: TwoPi = 6.28318530717958647693; { 2*Pi }

```

```

16665: PiDiv2      = 1.57079632679489661923; { Pi/2 }
16666: SqrtPi      = 1.77245385090551602730; { Sqrt(Pi) }
16667: Sqrt2Pi     = 2.50662827463100050242; { Sqrt(2*Pi) }
16668: InvSqrt2Pi  = 0.39894228040143267794; { 1/Sqrt(2*Pi) }
16669: LnSqrt2Pi   = 0.91893853320467274178; { Ln(Sqrt(2*Pi)) }
16670: Ln2PiDiv2   = 0.91893853320467274178; { Ln(2*Pi)/2 }
16671: Sqrt2       = 1.41421356237309504880; { Sqrt(2) }
16672: Sqrt2Div2   = 0.70710678118654752440; { Sqrt(2)/2 }
16673: Gold        = 1.61803398874989484821; { Golden Mean = (1 + Sqrt(5))/2 }
16674: CGold       = 0.38196601125010515179; { 2 - GOLD }
16675: MachEp      = 2.220446049250313E-16; { 2^(-52) }
16676: MaxNum      = 1.797693134862315E+308; { 2^1024 }
16677: MinNum      = 2.225073858507202E-308; { 2^(-1022) }
16678: MaxLog      = 709.7827128933840;
16679: MinLog      = -708.3964185322641;
16680: MaxFac      = 170;
16681: MaxGam      = 171.624376956302;
16682: MaxLgm      = 2.556348E+305;
16683: SingleCompareDelta = 1.0E-34;
16684: DoubleCompareDelta = 1.0E-280;
16685: { $IFDEF CLR }
16686: ExtendedCompareDelta = DoubleCompareDelta;
16687: { $ELSE }
16688: ExtendedCompareDelta = 1.0E-4400;
16689: { $ENDIF }
16690: Bytes1KB    = 1024;
16691: Bytes1MB    = 1024 * Bytes1KB;
16692: Bytes1GB    = 1024 * Bytes1MB;
16693: Bytes64KB   = 64 * Bytes1KB;
16694: Bytes64MB   = 64 * Bytes1MB;
16695: Bytes2GB    = 2 * LongWord(Bytes1GB);
16696: clBlack32   = $FF000000 );
16697: clDimGray32 = $FF3F3F3F );
16698: clGray32    = $FF7F7F7F );
16699: clLightGray32 = $FFBFBFBF );
16700: clWhite32   = $FFFFFFF );
16701: clMaroon32  = $FF7F0000 );
16702: clGreen32   = $FF007F00 );
16703: clOlive32   = $FF7F7F00 );
16704: clNavy32    = $FF00007F );
16705: clPurple32  = $FF7F007F );
16706: clTeal32    = $FF007F7F );
16707: clRed32     = $FFFF0000 );
16708: clLime32    = $FF00FF00 );
16709: clYellow32  = $FFFFFF00 );
16710: clBlue32    = $FF0000FF );
16711: clFuchsia32 = $FFFF00FF );
16712: clAqua32    = $FF00FFFF );
16713: clAliceBlue32 = $FFF0F8FF );
16714: clAntiqueWhite32 = $FFFAEBD7 );
16715: clAquamarine32 = $FFF7FFD4 );
16716: clAzure32   = $FFF0FFFF );
16717: clBeige32   = $FFF5F5DC );
16718: clBisque32  = $FFFFE4C4 );
16719: clBlancheDalmont32 = $FFFFEBCD );
16720: clBlueViolet32 = $FF8A2BE2 );
16721: clBrown32   = $FFA52A2A );
16722: clBurlyWood32 = $FFDEB887 );
16723: clCadetBlue32 = $FF5F9EA0 );
16724: clChartReuse32 = $FF7FFF00 );
16725: clChocolate32 = $FFD2691E );
16726: clCoral32   = $FFF7F500 );
16727: clCornFlowerBlue32 = $FF6495ED );
16728: clCornSilk32 = $FFFFFF8DC );
16729: clCrimson32 = $FFDC143C );
16730: clDarkBlue32 = $FF00008B );
16731: clDarkCyan32 = $FF008B8B );
16732: clDarkGoldenRod32 = $FFB8860B );
16733: clDarkGray32 = $FFA9A9A9 );
16734: clDarkGreen32 = $FF006400 );
16735: clDarkGrey32 = $FFA9A9A9 );
16736: clDarkKhaki32 = $FFBDB76B );
16737: clDarkMagenta32 = $FF8B008B );
16738: clDarkOliveGreen32 = $FF556B2F );
16739: clDarkOrange32 = $FFFF8C00 );
16740: clDarkOrchid32 = $FF9932CC );
16741: clDarkRed32 = $FF8B0000 );
16742: clDarkSalmon32 = $FFE9967A );
16743: clDarkSeaGreen32 = $FF8FBC8F );
16744: clDarkSlateBlue32 = $FF483D8B );
16745: clDarkSlateGray32 = $FF2F4F4F );
16746: clDarkSlateGrey32 = $FF2F4F4F );
16747: clDarkTurquoise32 = $FF00CED1 );
16748: clDarkViolet32 = $FF9400D3 );
16749: clDeepPink32 = $FFFF1493 );
16750: clDeepSkyBlue32 = $FF00BFFF );
16751: clDodgerBlue32 = $FF1E90FF );
16752: clFireBrick32 = $FF451551 );
16753: clFloralWhite32 = $FFFFFFF0 );

```

```
16754:    clGainsBoro32', $FFDCDCDC ));
16755:    clGhostWhite32', $FFF8F8FF ));
16756:    clGold32', $FFFFD700 ));
16757:    clGoldenRod32', $FFDAA520 ));
16758:    clGreenYellow32', $FFADFF2F ));
16759:    clGrey32', $FF808080 ));
16760:    clHoneyDew32', $FFF0FFF0 ));
16761:    clHotPink32', $FFF69B4 ));
16762:    clIndianRed32', $FFCD5C5C ));
16763:    clIndigo32', $FF4B0082 ));
16764:    clIvory32', $FFFFFFF0 ));
16765:    clKhaki32', $FFF0E68C ));
16766:    clLavender32', $FFE6E6FA ));
16767:    clLavenderBlush32', $FFFFF0F5 ));
16768:    clLawnGreen32', $FF7CFC00 ));
16769:    clLemonChiffon32', $FFFFFACD ));
16770:    clLightBlue32', $FFADD8E6 ));
16771:    clLightCoral32', $FFF08080 ));
16772:    clLightCyan32', $FFE0FFFF ));
16773:    clLightGoldenRodYellow32', $FFFAFAD2 ));
16774:    clLightGreen32', $FF90EE90 ));
16775:    clLightGrey32', $FFD3D3D3 ));
16776:    clLightPink32', $FFFB6C1 ));
16777:    clLightSalmon32', $FFFA07A ));
16778:    clLightSeagreen32', $FF20B2AA ));
16779:    clLightSkyblue32', $FF87CEFA ));
16780:    clLightSlategray32', $FF778899 ));
16781:    clLightSlategrey32', $FF778899 ));
16782:    clLightSteelblue32', $FFB0C4DE ));
16783:    clLightYellow32', $FFFFFFE0 ));
16784:    clLtGray32', $FFC0C0C0 ));
16785:    clMedGray32', $FFA0A0A4 ));
16786:    clDkGray32', $FF808080 ));
16787:    clMoneyGreen32', $FFC0DCC0 ));
16788:    clLegacySkyBlue32', $FFA6CAF0 ));
16789:    clCream32', $FFFFFBF0 ));
16790:    clLimeGreen32', $FF32CD32 ));
16791:    clLinen32', $FFFAF0E6 ));
16792:    clMediumAquamarine32', $FF66CDAA ));
16793:    clMediumBlue32', $FF0000CD ));
16794:    clMediumOrchid32', $FFBA55D3 ));
16795:    clMediumPurple32', $FF9370DB ));
16796:    clMediumSeaGreen32', $FF3CB371 ));
16797:    clMediumSlateBlue32', $FF7B68EE ));
16798:    clMediumSpringGreen32', $FF00FA9A ));
16799:    clMediumTurquoise32', $FF48D1CC ));
16800:    clMediumVioletRed32', $FFC71585 ));
16801:    clMidnightBlue32', $FF191970 ));
16802:    clMintCream32', $FFFF5FFFA ));
16803:    clMistyRose32', $FFFE4E1 ));
16804:    clMoccasin32', $FFFE4B5 ));
16805:    clNavajoWhite32', $FFFFDEAD ));
16806:    clOldLace32', $FFFD5E6 ));
16807:    clOliveDrab32', $FF6B8E23 ));
16808:    clOrange32', $FFFA500 ));
16809:    clOrangeRed32', $FFFF4500 ));
16810:    clOrchid32', $FFDA70D6 ));
16811:    clPaleGoldenRod32', $FFEE8AA ));
16812:    clPaleGreen32', $FF98FB98 ));
16813:    clPaleTurquoise32', $FFAFEEEE ));
16814:    clPaleVioletred32', $FFDB7093 ));
16815:    clPapayaWhip32', $FFFFEFD5 ));
16816:    clPeachPuff32', $FFFFDAB9 ));
16817:    clPeru32', $FFCD853F ));
16818:    clPlum32', $FFDDA0DD ));
16819:    clPowderBlue32', $FFB0E0E6 ));
16820:    clRosyBrown32', $FFBC8F8F ));
16821:    clRoyalBlue32', $FF4169E1 ));
16822:    clSaddleBrown32', $FF8B4513 ));
16823:    clSalmon32', $FFFA8072 ));
16824:    clSandyBrown32', $FFF4A460 ));
16825:    clSeaGreen32', $FF2E8B57 ));
16826:    clSeaShell32', $FFFFFF5EE ));
16827:    clSienna32', $FFA0522D ));
16828:    clSilver32', $FFC0C0C0 ));
16829:    clSkyblue32', $FF87CEEB ));
16830:    clSlateBlue32', $FF6A5ACD ));
16831:    clSlateGray32', $FF708090 ));
16832:    clSlateGrey32', $FF708090 ));
16833:    clSnow32', $FFFFFFAFA ));
16834:    clSpringgreen32', $FF00FF7F ));
16835:    clSteelblue32', $FF4682B4 ));
16836:    clTan32', $FFD2B48C ));
16837:    clThistle32', $FFD8BFD8 ));
16838:    clTomato32', $FFFF6347 ));
16839:    clTurquoise32', $FF40E0D0 ));
16840:    clViolet32', $FFEE82EE ));
16841:    clWheat32', $FFF5DEB3 ));
16842:    clWhitesmoke32', $FFF5F5F5 ));
```

```

16843:   clYellowgreen32', $FF9ACD32 ));
16844:   clTrWhite32', $FFFFFFF ));
16845:   clTrBlack32', $7F000000 ));
16846:   clTrRed32', $FFFF0000 ));
16847:   clTrGreen32', $7F00FF00 ));
16848:   clTrBlue32', $7F0000FF ));
16849:   // Fixed point math constants
16850:   FixedOne = $10000; FixedHalf = $7FFF;
16851:   FixedPI = Round(PI * FixedOne);
16852:   FixedToFloat = 1/FixedOne;
16853:
16854: Special Types
16855: *****
16856:   type Complex = record
16857:     X, Y : Float;
16858:   end;
16859:   type TVector = array of Float;
16860:   TIntVector = array of Integer;
16861:   TCompVector = array of Complex;
16862:   TBoolVector = array of Boolean;
16863:   TStrVector = array of String;
16864:   TMatrix = array of TVector;
16865:   TIntMatrix = array of TIntVector;
16866:   TCompMatrix = array of TCompVector;
16867:   TBoolMatrix = array of TBoolVector;
16868:   TStrMatrix = array of TStrVector;
16869:   TByteArray = array[0..32767] of byte; !
16870:   THexArray = array [0..15] of Char; // = '0123456789ABCDEF';
16871:   TBitmapStyle = (bsNormal, bsCentered, bsStretched);
16872:   T2StringArray = array of array of string;
16873:   T2IntegerArray = array of array of integer;
16874:   AddTypeS('INT_PTR', 'Integer
16875:   AddTypeS('LONG_PTR', 'Integer
16876:   AddTypeS('UINT_PTR', 'Cardinal
16877:   AddTypeS('ULONG_PTR', 'Cardinal
16878:   AddTypeS('DWORD_PTR', 'ULONG_PTR
16879:   TIntegerDynArray', 'array of Integer
16880:   TCardinalDynArray', 'array of Cardinal
16881:   TWordDynArray', 'array of Word
16882:   TSmallIntDynArray', 'array of SmallInt
16883:   TByteDynArray', 'array of Byte
16884:   TShortIntDynArray', 'array of ShortInt
16885:   TInt64DynArray', 'array of Int64
16886:   TLongWordDynArray', 'array of LongWord
16887:   TSingleDynArray', 'array of Single
16888:   TDoubleDynArray', 'array of Double
16889:   TBooleanDynArray', 'array of Boolean
16890:   TStringDynArray', 'array of string
16891:   TWideStringDynArray', 'array of WideString
16892:   TDynByteArray = array of Byte;
16893:   TDynShortintArray = array of Shortint;
16894:   TDynSmallintArray = array of Smallint;
16895:   TDynWordArray = array of Word;
16896:   TDynIntegerArray = array of Integer;
16897:   TDynLongintArray = array of Longint;
16898:   TDynCardinalArray = array of Cardinal;
16899:   TDynInt64Array = array of Int64;
16900:   TDynExtendedArray = array of Extended;
16901:   TDynDoubleArray = array of Double;
16902:   TDynSingleArray = array of Single;
16903:   TDynFloatArray = array of Float;
16904:   TDynPointerArray = array of Pointer;
16905:   TDynStringArray = array of string;
16906:   TSynSearchOption = (ssoMatchCase, ssoWholeWord, ssoBackwards,
16907:     ssoEntireScope, ssoSelectedOnly, ssoReplace, ssoReplaceAll, ssoPrompt);
16908:   TSynSearchOptions = set of TSynSearchOption;
16909:
16910:
16911:
16912: /** Project : Base Include RunTime Lib for maXbox *Name: pas_includebox.inc
16913: -----
16914: procedure drawPolygon(vPoints: TXYVector; cFrm: TForm);
16915: procedure drawPlot(vPoints: TXYVector; cFrm: TForm; vcolor: integer);
16916: procedure SaveCanvas(vCanvas: TCanvas; FileName: string);
16917: procedure SaveCanvas2(vCanvas: TCanvas; FileName: string);
16918: function CheckStringSum(vstring: string): integer;
16919: function HexToInt(HexNum: string): LongInt;
16920: function IntToBin(Int: Integer): String;
16921: function BinToInt(Binary: String): Integer;
16922: function HexToBin(HexNum: string): string; external2
16923: function BinToHex(Binary: String): string;
16924: function IntToFloat(i: Integer): double;
16925: function AddThousandSeparator(S: string; myChr: Char): string;
16926: function Max3(const X,Y,Z: Integer): Integer;
16927: procedure Swap(var X,Y: char); // faster without inline
16928: procedure ReverseString(var S: String);
16929: function CharToHexStr(Value: Char): string;
16930: function CharToUniCode(Value: Char): string;
16931: function Hex2Dec(Value: Str002): Byte;

```



```

16932: function HexStrCodeToStr(Value: string): string;
16933: function HexToStr(i: integer; value: string): string;
16934: function UniCodeToStr(Value: string): string;
16935: function CRC16(statement: string): string;
16936: function SearchForSubstrings(aStrList: TStringList; aSearchStr1, aSearchStr2: string): string;
16937: procedure SearchAndReplace(aStrList: TStringList; aSearchStr, aNewStr: string);
16938: procedure ShowInterfaces(myFile: string);
16939: function Fact2(av: integer): extended;
16940: function BoolToStr(B: Boolean): string;
16941: function GCD(x, y: LongInt): LongInt;
16942: function LCM(m,n: longint): longint;
16943: function GetASCII: string;
16944: function GetItemHeight(Font: TFont): Integer;
16945: function myPlaySound(s: pchar; flag, syncflag: integer): boolean;
16946: function myGetWindowsDirectory(lpBuffer: PChar; uSize: longword): longword;
16947: function getHINSTANCE: longword;
16948: function getHMODULE: longword;
16949: function GetASCII: string;
16950: function ByteIsOk(const AByte: string; var VB: Byte): boolean;
16951: function WordIsOk(const AWord: string; var VW: Word): boolean;
16952: function TwentyFourBitValueIsOk(const AValue: string; var VI: Integer): boolean;
16953: function LongIsOk(const ALong: string; var VC: Cardinal): boolean;
16954: function SafeStr(const s: string): string;
16955: function ExtractUrlPath(const FileName: string): string;
16956: function ExtractUrlName(const FileName: string): string;
16957: function IsInternet: boolean;
16958: function RotateLeft1Bit_u32( Value: uint32): uint32;
16959: procedure LinearRegression(const KnownY: array of Double; const KnownX: array of Double; NData: Int; var
  LF: TStLinEst; ErrorStats: Boolean);
16960: procedure getEnvironmentInfo;
16961: procedure AntiFreeze;
16962: function GetCPUSpeed: Double;
16963: function IsVirtualPcGuest: Boolean;
16964: function IsVmWareGuest: Boolean;
16965: procedure StartSerialDialog;
16966: function IsWow64: boolean;
16967: function IsWow64String(var s: string): Boolean;
16968: procedure StartThreadDemo;
16969: function RGB(R,G,B: Byte): TColor;
16970: function SendLn(amess: string): boolean;
16971: procedure maXbox;
16972: function AspectRatio(aWidth, aHeight: Integer): String;
16973: function wget(aURL, afile: string): boolean;
16974: procedure PrintList(Value: TStringList);
16975: procedure PrintImage(aValue: TBitmap; Style: TBitmapStyle);
16976: procedure getEnvironmentInfo;
16977: procedure AntiFreeze;
16978: function getBitmap(aphat: string): TBitmap;
16979: procedure ShowMessageBig(const aText: string);
16980: function YesNoDialog(const ACaption, AMsg: string): boolean;
16981: procedure SetArrayLength2String(arr: T2StringArray; asize1, asize2: integer);
16982: procedure SetArrayLength2Integer(arr: T2IntegerArray; asize1, asize2: integer);
16983: //function myStrToBytes(const Value: String): TBytes;
16984: //function myBytesToStr(const Value: TBytes): String;
16985: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
16986: function getBitmap(aphat: string): TBitmap;
16987: procedure ShowMessageBig(const aText: string);
16988: function StrToBytes(const Value: String): TBytes;
16989: function BytesToStr(const Value: TBytes): String;
16990: function SaveAsExcelFile(AGrid: TStringGrid; ASheetName, AFileName: string; open: boolean): Boolean;
16991: function ReverseDNSLookup(const IPAdrs: String; const DNSServer: String; Timeout, Retries: Int; var
  HostName: String): Bool;
16992: function FindInPaths(const fileName, paths: String): String;
16993: procedure initHexArray(var hexn: THexArray);
16994: function josephusG(n,k: integer; var graphout: string): integer;
16995: function isPowerof2(num: int64): boolean;
16996: function powerOf2(exponent: integer): int64;
16997: function getBigPI: string;
16998: procedure MakeSound(Frequency{Hz}, Duration{mSec}: Integer; Volume: TVolumeLevel; savefilePath: string);
16999: function GetASCIILine: string;
17000: procedure MakeComplexSound(N: integer{stream # to use}; freqlist: TStringList; Duration{mSec}: Integer;
  pinknoise: boolean; shape: integer; Volume: TVolumeLevel);
17001:
17002: procedure SetComplexSoundElements(freqedt, Phaseedt, AmpEdt, WaveGrp: integer);
17003: procedure AddComplexSoundObjectToList(newf, newp, newa, news: integer; freqlist: TStringList);
17004: function mapfunc(ax, in_min, in_max, out_min, out_max: integer): integer;
17005: function mapmax(ax, in_min, in_max, out_min, out_max: integer): integer;
17006: function isKeyPressed: boolean;
17007: function KeyPress: boolean;
17008: procedure StrSplitP(const Delimiter: Char; Input: string; const Strings: TStringList);
17009: function ReadReg(Base: HKEY; KeyName, ValueName: string): string;
17010: function ReadRegistry(Base: HKEY; KeyName, ValueName: string): string;
17011: function GetOSName: string;
17012: function GetOSVersion: string;
17013: function GetOSNumber: string;
17014: function getEnvironmentString: string;
17015: procedure StrReplace(var Str: String; Old, New: String);
17016: procedure SendEmail(mFrom, mTo, mSubject, mBody, mAttachment: variant);
17017: function getTeamViewerID: string;
17018: procedure RecurseDirectory(Dir: String; IncludeSubs: boolean; callback: TFileCallbackProcedure);

```

```

17019: Procedure RecurseDirectory2(Dir : String; IncludeSubs : boolean);
17020: procedure WinInet_HttpGet(const Url: string; Stream:TStream);
17021: procedure GetQrCode2(Width,Height: Word;Correct_Level:string;const Data:string;apath:string);
17022: function StartSocketService: Boolean;
17023: procedure StartSocketServiceForm;
17024: function GetFileList(FileList: TStringlist; apath: string): TStringlist;
17025: function GetFileList1(apath: string): TStringlist;
17026: procedure LetFileList(FileList: TStringlist; apath: string);
17027: procedure StartWeb(aurl: string);
17028: function GetTodayFiles(startdir, amask: string): TStringlist;
17029: function PortTCPIsOpen(dwPort : Word; ipAddressStr: String): boolean;
17030: function JavahashCode(val: string): Integer;
17031: procedure PostKeyEx32(key: Word; const shift: TShiftState; specialkey: Boolean);
17032: procedure SaveBytesToFile2(const Data: Sysutils.TBytes; const FileName: string);
17033: Procedure HideWindowForSeconds(secs: integer); {//3 seconds}
17034: Procedure HideWindowForSeconds2(secs: integer; apphandle, aself: TForm); {//3 seconds}
17035: Procedure ConvertToGray(Cnv: TCanvas);
17036: function GetFileDate(aFile:string; aWithTime:Boolean):string;
17037: procedure ShowMemory;
17038: function ShowMemory2: string;
17039: function getHostIP: string;
17040: procedure ShowBitmap(bmap: TBitmap);
17041: function GetOsVersionInfo: TOSVersionInfo; //thx to wischnewski
17042:
17043:
17044: // News of 3.9.8 up
17045: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
17046: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
17047: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
17048: JvChart - TjvChart Component - 2009 Public
17049: MemoryLeakReport in ini-file (MEMORYREPORT=Y)
17050: PerlRegEx PCRE obj lib included, Perl & Python Syntax Editor, bitbox3 logic example
17051: TAdoQuery.SQL.Add() fixed, ShLwAPI extensions, Indy HTTPHeader Extensions
17052: DMath DLL included incl. Demos
17053: Interface Navigator menu/View/Intf Navigator
17054: Unit Explorer menu/Debug/Units Explorer
17055: EKON 16 Slides ..\maxbox3\docs\utils Excel Export maxXcel
17056: Tutorial 19 WinCOM with Arduino Tutorial 20 RegEx Coding
17057: Script History to 9 Files WebServer light /Options/Addons/WebServer
17058: Full Text Finder, JVSImLogic Simulator Package
17059: Halt-Stop Program in Menu, WebServer2, Stop Event ,
17060: Conversion Routines, Prebuild Forms, CodeSearch
17061: Halt-Stop Program in Menu, WebServer2, Stop Event Recompile,
17062: Conversion Routines, Prebuild Forms, more RCData, DebugOutString
17063: CodeSearchEngine to search code patterns in /examples <Ctrl F3>
17064: JvChart - TjvChart Component - 2009 Public, mXGames, JvgXMLSerializer, TjvPaintFX
17065: Compress-Decompress Zip, Services Tutorial22, Synopse framework, PFDLib
17066: SynEdit API, Macro, Macro Recorder, DLL Spy, Configuration Tutorial
17067: IDE Reflection API, Session Service Shell S3
17068: additional SynEdit API, isKeyPressed Routine, Bookmarks, OpenToolsAPI Catalog (OTAC)
17069: Class TMonitor, Configuration Tutorial maxbox_starter25.pdf, Chess.dll Game
17070: arduino map() function, PMRandom Generator
17071: StBarCode Lib, StreamReaderClass, BarCode Package, Astro Package
17072: more ShellAPI, add 32 more units, Simulated Annealing, GenAlgo
17073: REST Test Lib, Multilang Component, Forth Interpreter
17074: New Macros, Sendmail (instant email), DevCUnits, Tetris Addon
17075: DCOM, MDAC, MIDI, TLS support, Posmarks, Utils Addon
17076: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
17077: Routines for LaTeX/PS, Utils Addon, Indy Package3, TAR Archive, @Callbacks
17078: First LCL of Lazarus, CmdLine API, ToDo List, 36 more Units preCompiled
17079: QRCode Service, add more CFunctions like CDateTime of Synapse
17080: Gamma Functions, IndyPackage4, HotLog Threadable, FormTemplateLibrary FTL
17081: Nonlinear regression, ADO Workbench Addon, Assign fixing, IntfNavigator fixing, Applet
17082: 30 more Units preCompiled, QRCode Indy Service, more CFunctions like CFill or SRand
17083: RestartDialog, RTF, SQL Scanner, RichEdit, 15 more Units
17084: Tool Section, SOAP Tester, Hot Log Logger2, TCPPortScan, 28 more Units
17085: BOLD Package, Indy Package5, maTRIX. MATHEMAX
17086: SPS Utils WDOS, Plc BitBus (Petrinet), 40 more units
17087: emax layers: system-package-component-unit-class-function-block
17088: HighPrecision Timers, Indy Package6, AutoDetect, UltraForms
17089: Reversi, GOL, bugfixing, 8 more units, Tutorial 24 Clean Code
17090: Tutorial 18_3 RGB LED, OpenGL Geometry, maxpix, statictext
17091: OpenGL Game Demo: ..Options/Add Ons/Reversi
17092: IBUtils Refactor, InterBase Package, DotNet Routines (JvExControls)
17093: add 31 units, mX4 Introduction Paper, more Socket&Streams, ShortString Routines
17094: 7% performance gain (hot spot profiling)
17095: PEP -Pascal Education Program , GSM Module, CGI, PHP Runner
17096: add 42 + 22 (64 units), memcached database, autobookmark, Alcino PAC, IPC Lib
17097: Orpheus PAC, AsyncFree Library advapi32 samples, FirebirdExp+MySQL units
17098: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
17099:
17100: add routines in 3.9.7.5
17101: 097: procedure RIRRegister_BarCodeScanner_Routines(S: TPSExec);
17102: 996: procedure RIRRegister_DBCtrls_Routines(S: TPSExec);
17103: 069: procedure RIRRegister_IdStrings_Routines(S: TPSExec);
17104: 516: procedure RIRRegister_JclMultimedia_Routines(S: TPSExec);
17105: 215: procedure RIRRegister_PNGLoader_Routines(S: TPSExec);
17106: 374: procedure RIRRegister_SerDlgs_Routines(S: TPSExec);
17107: 777: procedure RIRRegister_LinarBitmap_Routines(S: TPSExec);

```

```

17108:
17109: ////////////////////////////////////////////////// TestUnits ///////////////////////////////////
17110: SelftestPEM;
17111: SelfTestCFundamentUtils;
17112: SelfTestCFileUtils;
17113: SelfTestCDateTime;
17114: SelfTestCTimer;
17115: SelfTestCRandom;
17116: Test with e.g.: Assert(PathHasDriveLetter('A:'), 'PathHasDriveLetter
17117: Assert(WinPathToUnixPath('\c\d.f') = '/c/d.f', 'WinPathToUnixPath
17118:
17119: // Note: There's no need for installing a client certificate in the
17120: // webbrowser. The server asks the webbrowser to send a certificate but
17121: // if nothing is installed the software will work because the server
17122: // doesn't check to see if a client certificate was supplied. If you want you can install:
17123: //
17124: // file: c_cacert.p12
17125: // password: c_cakey
17126:
17127: TGraphicControl = class(TControl)
17128: private
17129: FCanvas: TCanvas;
17130: procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
17131: protected
17132: procedure Paint; virtual;
17133: property Canvas: TCanvas read FCanvas;
17134: public
17135: constructor Create(AOwner: TComponent); override;
17136: destructor Destroy; override;
17137: end;
17138:
17139: TCustomControl = class(TWinControl)
17140: private
17141: FCanvas: TCanvas;
17142: procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
17143: protected
17144: procedure Paint; virtual;
17145: procedure PaintWindow(DC: HDC); override;
17146: property Canvas: TCanvas read FCanvas;
17147: public
17148: constructor Create(AOwner: TComponent); override;
17149: destructor Destroy; override;
17150: end;
17151: RegisterPublishedProperties;
17152: ('ONCHANGE', 'TNotifyEvent', iptrw);
17153: ('ONCLICK', 'TNotifyEvent', iptrw);
17154: ('ONDBLCLICK', 'TNotifyEvent', iptrw);
17155: ('ONENTER', 'TNotifyEvent', iptrw);
17156: ('ONEXIT', 'TNotifyEvent', iptrw);
17157: ('ONKEYDOWN', 'TKeyEvent', iptrw);
17158: ('ONKEYPRESS', 'TKeyPressEvent', iptrw);
17159: ('ONMOUSEDOWN', 'TMouseEvent', iptrw);
17160: ('ONMOUSEMOVE', 'TMouseMoveEvent', iptrw);
17161: ('ONMOUSEUP', 'TMouseEvent', iptrw);
17162: //*****
17163: // To stop the while loop, click on Options/Show Include (boolean switch)!
17164: Control a loop in a script with a form event:
17165: IncludeON; //control the while loop
17166: while maxform1.ShowInclude1.checked do begin //menu event Options/Show Include
17167:
17168: //-----
17169: //*****mX4 ini-file Configuration*****
17170: //-----
17171: using config file maxboxdef.ini menu/Help/Config File
17172:
17173: /*** Definitions for maXbox mX3 ***/
17174: [FORM]
17175: LAST_FILE=E:\maXbox\maxbox3\examples\140_drive_typedemo.txt //history up to 10 files
17176: FONTSIZE=14
17177: EXTENSION=txt
17178: SCREENX=1386
17179: SCREENY=1077
17180: MEMHEIGHT=350
17181: PRINTFONT=Courier New //GUI Settings
17182: LINENUMBERS=Y //line numbers at gutter in editor at left side
17183: EXCEPTIONLOG=Y //store excerpts and success in 2 log files see below! - menu Debug/Show Last Exceptions
17184: EXECUTESHELL=Y //prevents execution of ExecuteShell() or ExecuteCommand()
17185: BOOTSCRIPT=Y //enabling load a boot script
17186: MEMORYREPORT=Y //shows memory report on closing maXbox
17187: MACRO=Y //expand macros (see below) in code e.g. #path:E:\maxbox\maxbox3\docs\
17188: NAVIGATOR=N //shows function list at the right side of editor
17189: NAVWIDTH=350 //width of the right side interface list <CTRL L>
17190: AUTOBOOKMARK=Y //sets on all functions a bookmark to jump
17191: [WEB]
17192: IPPORT=8080 //for internal webserver - menu /Options/Add Ons/WebServer
17193: IPHOST=192.168.1.53
17194: ROOTCERT=filepathY
17195: SCERT=filepathY
17196: RSAKEY=filepathY

```

```

17197: VERSIONCHECK=Y
17198:
17199: using Logfile: maxboxlog.log , Exceptionlogfile: maxboxerrorlog.txt
17200:
17201: Also possible to set report memory in script to override ini setting
17202: procedure Set_ReportMemoryLeaksOnShutdown(abo: boolean)
17203:
17204:
17205: After Change the ini file you can reload the file with ../Help/Config Update
17206:
17207: //-----mX4 maildef.ini ini-file Configuration-----
17208: //*****mX4 maildef.ini ini-file Configuration*****
17209: //-----
17210: /*** Definitions for maxMail ***/
17211: //sendemail, HOST=mail.hover.com, PORT=465 (SSL)
17212: [MAXMAIL]
17213: HOST=getmail.softwareschule.ch
17214: USER=mailusername
17215: PASS=password
17216: PORT=110
17217: SSL=Y
17218: BODY=Y
17219: LAST=5
17220:
17221: ADO Connection String:
17222: Provider=MSDASQL;DSN=mx3base;Uid=sa;Pwd=admin
17223:
17224: OpenSSL Lib: unit ssl_openssl_lib;
17225: {$IFDEF CIL}
17226: const
17227:   {$IFDEF LINUX}
17228:     DLLSSLName = 'libssl.so';
17229:     DLLUtilName = 'libcrypto.so';
17230:   {$ELSE}
17231:     DLLSSLName = 'ssleay32.dll';
17232:     DLLUtilName = 'libeay32.dll';
17233:   {$ENDIF}
17234: {$ELSE}
17235: var
17236:   {$IFDEF MSWINDOWS}
17237:     {$IFDEF DARWIN}
17238:       DLLSSLName: string = 'libssl.dylib';
17239:       DLLUtilName: string = 'libcrypto.dylib';
17240:     {$ELSE}
17241:       DLLSSLName: string = 'libssl.so';
17242:       DLLUtilName: string = 'libcrypto.so';
17243:     {$ENDIF}
17244:   {$ELSE}
17245:     DLLSSLName: string = 'ssleay32.dll';
17246:     DLLSSLName2: string = 'libssl32.dll';
17247:     DLLUtilName: string = 'libeay32.dll';
17248:   {$ENDIF}
17249: {$ENDIF}
17250:
17251:
17252:
17253: //-----mX4 Macro Tags -----
17254: //*****mX4 Macro Tags *****
17255: //-----
17256:
17257: asm #name #hostmAPSN2APSN2111e, #head,max: APSN21: 04.01.2014 19:05:50
E:\maxbox\maxbox3\docs\maxbox_extract_funclist399.txt end
17258:
17259: //Tag Macros
17260:
17261: asm #name, #date, #host, #path, #file, #head, #sign #tech #net end
17262:
17263: //Tag Macros
17264: 10188: SearchAndCopy(memol.lines, '#name', getUsernameWin, 11);
17265: 10189: SearchAndCopy(memol.lines, '#date', datetimetoStr(now), 11);
17266: 10190: SearchAndCopy(memol.lines, '#host', getComputernameWin, 11);
17267: 10191: SearchAndCopy(memol.lines, '#path', fpath, 11);
17268: 10192: SearchAndCopy(memol.lines, '#file', fname, 11);
17269: 10199: SearchAndCopy(memol.lines, '#fils', fname + ' +SHA1(Act_Filename), 11);
17270: 10193: SearchAndCopy(memol.lines, '#locs', intToStr(getCodeEnd), 11);
17271: 10194: SearchAndCopy(memol.lines, '#perf', perfTime, 11);
17272: 10195: SearchAndCopy(memol.lines, '#sign', Format('%s: %s: %s',
17273:   [getUsernameWin, getComputernameWin, datetimetoStr(now),
17274:   SearchAndCopy(memol.lines, '#head', Format('%s: %s: %s %s ',
17275:   [getUsernameWin, getComputernameWin, datetimetoStr(now), Act_Filename]],11);
17276:   [getUsernameWin, getComputernameWin, datetimetoStr(now), Act_Filename]],11);
17277: 10198: SearchAndCopy(memol.lines, '#tech', Format('perf: %s threads: %d %s %s',
17278:   [perfTime, numprocessthreads, getIPAddress(getComputerNameWin), timetoStr(time), mbversion]), 11);
17279: 10298: SearchAndCopy(memol.lines, '#net', Format('DNS: %s; local IPs: %s; local IP: %s',
17280:   [getDNS, GetLocalIPs, getIPAddress(getComputerNameWin)], 10);
17281:
17282: // #tech!perf: 0:0:29.297 threads: 3 192.168.174.1 19:26:30
17283:
17284: //Replace Macros

```



```

17285: SearchAndCopy(memol.lines, '<TIME>', timetoStr(time), 6);
17286: SearchAndCopy(memol.lines, '<DATE>', datetoStr(date), 6);
17287: SearchAndCopy(memol.lines, '<PATH>', fpath, 6);
17288: SearchAndCopy(memol.lines, '<EXEPATH>', ExePath, 9);
17289: SearchAndCopy(memol.lines, '<FILE>', fname, 6);
17290: SearchAndCopy(memol.lines, '<SOURCE>', ExePath+'Source', 8);
17291:
17292: 10198: SearchAndCopy(memol.lines, '#tech'perf: threads: 2 192.168.1.53 19:05:50 3.9.9.84
17293: [perftime,numprocessthreads,getIPAddress(getComputerNameWin),timetoStr(time),mbversion]], 11);
17294: //#tech!84perf: threads: 2 192.168.1.53 19:05:50 3.9.9.84
17295: SearchAndCopy(memol.lines, 'maxbox_extract_funclist399.txt
17296:
17297: -----
17298: *****mX4 ToDo List Tags ../Help/ToDo List*****
17299: -----
17300:
17301: while I < sl.Count do begin
17302: // if MatchesMask(sl[I], '/*/? TODO ([a-z0-9_]*#[1-9]#)*:~') then
17303: if MatchesMask(sl[I], '/*/? TODO (?*#?#)*:~') then
17304: BreakupToDo(Filename, sl, I, 'TODO', True, True) // full info TODO
17305: else if MatchesMask(sl[I], '/*/? DONE (?*#?#)*:~') then
17306: BreakupToDo(Filename, sl, I, 'DONE', True, True) // full info DONE
17307: else if MatchesMask(sl[I], '/*/? TODO (?*#?#)*:~') then
17308: BreakupToDo(Filename, sl, I, 'TODO', False, True) // only priority info TODO
17309: else if MatchesMask(sl[I], '/*/? DONE (?*#?#)*:~') then
17310: BreakupToDo(Filename, sl, I, 'DONE', False, True) // only priority info DONE
17311: else if MatchesMask(sl[I], '/*/?*TODO*:~') then
17312: BreakupToDo(Filename, sl, I, 'TODO', False, False) // custom TODO
17313: else if MatchesMask(sl[I], '/*/?*DONE*:~') then
17314: BreakupToDo(Filename, sl, I, 'DONE', False, False); // custom DONE
17315: Inc(I);
17316: end;
17317:
17318:
17319: -----
17320: *****mX4 Public Tools API *****
17321: -----
17322: file : unit uPSI_fMain.pas; {SOTAP} Open Tools API Catalog
17323: // Those functions concern the editor and preprocessor, all of the IDE
17324: Example: Call it with maxform1.InfolClick(self)
17325: Note: Call all Methods with maxForm1., e.g.:
17326: maxForm1.ShellStyle1Click(self);
17327:
17328: procedure SIRegister_fMain(CL: TPSPascalCompiler);
17329: begin
17330: Const('BYTECODE','String').SetString( 'bytecode.txt
17331: Const('PSTEXT','String').SetString( 'PS Scriptfiles (*.txt)|*.TXT
17332: Const('PSMODEL','String').SetString( 'PS Modelfiles (*.uc)|*.UC
17333: Const('PSPASCAL','String').SetString( 'PS Pascalfiles (*.pas)|*.PAS
17334: Const('PSINC','String').SetString( 'PS Includes (*.inc)|*.INC
17335: Const('DEFFILENAME','String').SetString( 'firstdemo.txt
17336: Const('DEFINIFILE','String').SetString( 'maxboxdef.ini
17337: Const('EXCEPTLOGFILE','String').SetString( 'maxboxerrorlog.txt
17338: Const('ALLFUNCTIONSLIST','String').SetString( 'upsi_allfunctionslist.txt
17339: Const('ALLFUNCTIONSLISTPDF','String').SetString( 'maxbox_functions_all.pdf
17340: Const('ALLOBJECTSLIST','String').SetString( 'docs\VCL.pdf
17341: Const('ALLRESOURCELIST','String').SetString( 'docs\upsi_allresourcelist.txt
17342: Const('ALLUNITLIST','String').SetString( 'docs\maxbox3_9.xml');
17343: Const('INCLUDEBOX','String').SetString( 'pas_includebox.inc
17344: Const('BOOTSCRIPT','String').SetString( 'maxbootscript.txt
17345: Const('MBVERSION','String').SetString( '3.9.9.91
17346: Const('VERSION','String').SetString( '3.9.9.91
17347: Const('MBVER','String').SetString( '399
17348: Const('MBVERI','Integer').SetInt(399);
17349: Const('MBVERIALL','Integer').SetInt(39991);
17350: Const('EXENAME','String').SetString( 'maXbox3.exe
17351: Const('MXSITE','String').SetString( 'http://www.softwareschule.ch/maxbox.htm
17352: Const('MXVERSIONFILE','String').SetString( 'http://www.softwareschule.ch/maxvfile.txt
17353: Const('MXINTERNETCHECK','String').SetString( 'www.ask.com
17354: Const('MXMAIL','String').SetString( 'max@kleiner.com
17355: Const('TAB','Char').SetString( #09);
17356: Const('CODECOMPLETION','String').SetString( 'bds_delphi.dci
17357: SIRegister_TMaxForm1(CL);
17358: end;
17359:
17360: with FindClass('TForm'),'TMaxForm1' do begin
17361: memo2', 'TMemo', iptrw);
17362: memol', 'TSynMemo', iptrw);
17363: CB1SCList', 'TComboBox', iptrw);
17364: mxNavigator', 'TComboBox', iptrw);
17365: IPHost', 'string', iptrw);
17366: IPPort', 'integer', iptrw);
17367: COMPort', 'integer', iptrw); //3.9.6.4
17368: Splitter1', 'TSplitter', iptrw);
17369: PSScript', 'TPSScript', iptrw);
17370: PS3DllPlugin', 'TPSDllPlugin', iptrw);
17371: MainMenu1', 'TMainMenu', iptrw);
17372: Program1', 'TMenuItem', iptrw);
17373: Compile1', 'TMenuItem', iptrw);

```

```
17374: Files1', 'TMenuItem', iptrw);
17375: open1', 'TMenuItem', iptrw);
17376: Save2', 'TMenuItem', iptrw);
17377: Options1', 'TMenuItem', iptrw);
17378: Savebefore1', 'TMenuItem', iptrw);
17379: Largefont1', 'TMenuItem', iptrw);
17380: sBytecodel', 'TMenuItem', iptrw);
17381: Saveas3', 'TMenuItem', iptrw);
17382: Clear1', 'TMenuItem', iptrw);
17383: Slinenumber1', 'TMenuItem', iptrw);
17384: About1', 'TMenuItem', iptrw);
17385: Search1', 'TMenuItem', iptrw);
17386: SynPasSyn1', 'TSynPasSyn', iptrw);
17387: memol', 'TSynMemo', iptrw);
17388: SynEditSearch1', 'TSynEditSearch', iptrw);
17389: WordWrap1', 'TMenuItem', iptrw);
17390: XPManifest1', 'TXPManifest', iptrw);
17391: SearchNext1', 'TMenuItem', iptrw);
17392: Replacel', 'TMenuItem', iptrw);
17393: PSImport_Controls1', 'TPSImport_Controls', iptrw);
17394: PSImport_Classes1', 'TPSImport_Classes', iptrw);
17395: ShowIncludel', 'TMenuItem', iptrw);
17396: SynEditPrint1', 'TSynEditPrint', iptrw);
17397: Printout1', 'TMenuItem', iptrw);
17398: mnPrintColors1', 'TMenuItem', iptrw);
17399: dlgFilePrint', 'TPrintDialog', iptrw);
17400: dlgPrintFont1', 'TFontDialog', iptrw);
17401: mnuPrintFont1', 'TMenuItem', iptrw);
17402: Includel', 'TMenuItem', iptrw);
17403: CodeCompletionList1', 'TMenuItem', iptrw);
17404: IncludeList1', 'TMenuItem', iptrw);
17405: ImageList1', 'TImageList', iptrw);
17406: ImageList2', 'TImageList', iptrw);
17407: CoolBar1', 'TCoolBar', iptrw);
17408: ToolBar1', 'TToolBar', iptrw);
17409: tbtnLoad', 'TToolButton', iptrw);
17410: ToolButton2', 'TToolButton', iptrw);
17411: tbtnFind', 'TToolButton', iptrw);
17412: tbtnCompile', 'TToolButton', iptrw);
17413: tbtnTrans', 'TToolButton', iptrw);
17414: tbtnUseCase', 'TToolButton', iptrw); //3.8
17415: toolbtnTutorial', 'TToolButton', iptrw);
17416: tbtn6res', 'TToolButton', iptrw);
17417: ToolButton5', 'TToolButton', iptrw);
17418: ToolButton1', 'TToolButton', iptrw);
17419: ToolButton3', 'TToolButton', iptrw);
17420: statusBar1', 'TStatusBar', iptrw);
17421: SaveOutput1', 'TMenuItem', iptrw);
17422: ExportClipboard1', 'TMenuItem', iptrw);
17423: Closes1', 'TMenuItem', iptrw);
17424: Manual1', 'TMenuItem', iptrw);
17425: About2', 'TMenuItem', iptrw);
17426: loadLastfile1', 'TMenuItem', iptrw);
17427: imglogo', 'TImage', iptrw);
17428: cedebug', 'TPSScriptDebugger', iptrw);
17429: debugPopupMenu1', 'TPopupMenu', iptrw);
17430: BreakPointMenu', 'TMenuItem', iptrw);
17431: Decompile1', 'TMenuItem', iptrw);
17432: StepIntol', 'TMenuItem', iptrw);
17433: StepOut1', 'TMenuItem', iptrw);
17434: Reset1', 'TMenuItem', iptrw);
17435: DebugRun1', 'TMenuItem', iptrw);
17436: PSImport_ComObj1', 'TPSImport_ComObj', iptrw);
17437: PSImport_StdCtrls1', 'TPSImport_StdCtrls', iptrw);
17438: PSImport_Forms1', 'TPSImport_Forms', iptrw);
17439: PSImport_DateUtils1', 'TPSImport_DateUtils', iptrw);
17440: tutorial4', 'TMenuItem', iptrw);
17441: ExporttoClipboard1', 'TMenuItem', iptrw);
17442: ImportfromClipboard1', 'TMenuItem', iptrw);
17443: N4', 'TMenuItem', iptrw);
17444: N5', 'TMenuItem', iptrw);
17445: N6', 'TMenuItem', iptrw);
17446: ImportfromClipboard2', 'TMenuItem', iptrw);
17447: tutorial1', 'TMenuItem', iptrw);
17448: N7', 'TMenuItem', iptrw);
17449: ShowSpecChars1', 'TMenuItem', iptrw);
17450: OpenDirectory1', 'TMenuItem', iptrw);
17451: procMess', 'TMenuItem', iptrw);
17452: tbtnUseCase', 'TToolButton', iptrw);
17453: ToolButton7', 'TToolButton', iptrw);
17454: EditFont1', 'TMenuItem', iptrw);
17455: UseCase1', 'TMenuItem', iptrw);
17456: tutorial21', 'TMenuItem', iptrw);
17457: OpenUseCase1', 'TMenuItem', iptrw);
17458: PSImport_DBI', 'TPSImport_DB', iptrw);
17459: tutorial31', 'TMenuItem', iptrw);
17460: SynHTMLSyn1', 'TSynHTMLSyn', iptrw);
17461: HTMLSyntax1', 'TMenuItem', iptrw);
17462: ShowInterfaces1', 'TMenuItem', iptrw);
```

```
17463: Tutorial5', 'TMenuItem', iptrw);
17464: AllFunctionsList1', 'TMenuItem', iptrw);
17465: ShowLastException1', 'TMenuItem', iptrw);
17466: PlayMP31', 'TMenuItem', iptrw);
17467: SynTeXSyn1', 'TSynTeXSyn', iptrw);
17468: texSyntax1', 'TMenuItem', iptrw);
17469: N8', 'TMenuItem', iptrw);
17470: GetEmails1', 'TMenuItem', iptrw);
17471: SynCppSyn1', 'TSynCppSyn', iptrw);
17472: CSyntax1', 'TMenuItem', iptrw);
17473: Tutorial6', 'TMenuItem', iptrw);
17474: New1', 'TMenuItem', iptrw);
17475: AllObjectsList1', 'TMenuItem', iptrw);
17476: LoadBytecode1', 'TMenuItem', iptrw);
17477: CipherFile1', 'TMenuItem', iptrw);
17478: N9', 'TMenuItem', iptrw);
17479: N10', 'TMenuItem', iptrw);
17480: Tutorial11', 'TMenuItem', iptrw);
17481: Tutorial71', 'TMenuItem', iptrw);
17482: UpdateService1', 'TMenuItem', iptrw);
17483: PascalSchool1', 'TMenuItem', iptrw);
17484: Tutorial81', 'TMenuItem', iptrw);
17485: DelphiSite1', 'TMenuItem', iptrw);
17486: Output1', 'TMenuItem', iptrw);
17487: TerminalStyle1', 'TMenuItem', iptrw);
17488: ReadOnly1', 'TMenuItem', iptrw);
17489: ShellStyle1', 'TMenuItem', iptrw);
17490: BigScreen1', 'TMenuItem', iptrw);
17491: Tutorial91', 'TMenuItem', iptrw);
17492: SaveOutput2', 'TMenuItem', iptrw);
17493: N11', 'TMenuItem', iptrw);
17494: SaveScreenshot', 'TMenuItem', iptrw);
17495: Tutorial101', 'TMenuItem', iptrw);
17496: SQLSyntax1', 'TMenuItem', iptrw);
17497: SynSQLSyn1', 'TSynSQLSyn', iptrw);
17498: Console1', 'TMenuItem', iptrw);
17499: SynXMLSyn1', 'TSynXMLSyn', iptrw);
17500: XMLSyntax1', 'TMenuItem', iptrw);
17501: ComponentCount1', 'TMenuItem', iptrw);
17502: NewInstance1', 'TMenuItem', iptrw);
17503: toolbtnTutorial', 'TToolButton', iptrw);
17504: Memory1', 'TMenuItem', iptrw);
17505: SynJavaSyn1', 'TSynJavaSyn', iptrw);
17506: JavaSyntax1', 'TMenuItem', iptrw);
17507: SyntaxCheck1', 'TMenuItem', iptrw);
17508: Tutorial10Statistics1', 'TMenuItem', iptrw);
17509: ScriptExplorer1', 'TMenuItem', iptrw);
17510: FormOutput1', 'TMenuItem', iptrw);
17511: ArduinoDump1', 'TMenuItem', iptrw);
17512: AndroidDump1', 'TMenuItem', iptrw);
17513: GotoEnd1', 'TMenuItem', iptrw);
17514: AllResourceList1', 'TMenuItem', iptrw);
17515: ToolButton4', 'TToolButton', iptrw);
17516: tbtn6res', 'TToolButton', iptrw);
17517: Tutorial11Forms1', 'TMenuItem', iptrw);
17518: Tutorial12SQL1', 'TMenuItem', iptrw);
17519: ResourceExplore1', 'TMenuItem', iptrw);
17520: Infol', 'TMenuItem', iptrw);
17521: N12', 'TMenuItem', iptrw);
17522: CryptoBox1', 'TMenuItem', iptrw);
17523: Tutorial13Ciphering1', 'TMenuItem', iptrw);
17524: CipherFile2', 'TMenuItem', iptrw);
17525: N13', 'TMenuItem', iptrw);
17526: ModulesCount1', 'TMenuItem', iptrw);
17527: AddOns2', 'TMenuItem', iptrw);
17528: N4GewinntGame1', 'TMenuItem', iptrw);
17529: DocuforAddOns1', 'TMenuItem', iptrw);
17530: Tutorial14Async1', 'TMenuItem', iptrw);
17531: Lessons15Review1', 'TMenuItem', iptrw);
17532: SynPHPSyn1', 'TSynPHPSyn', iptrw);
17533: PHPSyntax1', 'TMenuItem', iptrw);
17534: Breakpoint1', 'TMenuItem', iptrw);
17535: SerialRS2321', 'TMenuItem', iptrw);
17536: N14', 'TMenuItem', iptrw);
17537: SynCSSyn1', 'TSynCSSyn', iptrw);
17538: CSyntax2', 'TMenuItem', iptrw);
17539: Calculator1', 'TMenuItem', iptrw);
17540: tbtnSerial', 'TToolButton', iptrw);
17541: ToolButton8', 'TToolButton', iptrw);
17542: Tutorial151', 'TMenuItem', iptrw);
17543: N15', 'TMenuItem', iptrw);
17544: N16', 'TMenuItem', iptrw);
17545: ControlBar1', 'TControlBar', iptrw);
17546: ToolBar2', 'TToolBar', iptrw);
17547: BtnOpen', 'TToolButton', iptrw);
17548: BtnSave', 'TToolButton', iptrw);
17549: BtnPrint', 'TToolButton', iptrw);
17550: BtnColors', 'TToolButton', iptrw);
17551: btnClassReport', 'TToolButton', iptrw);
```

```

17552: BtnRotateRight', 'TToolButton', iptrw);
17553: BtnFullSize', 'TToolButton', iptrw);
17554: BtnFitToWindowSize', 'TToolButton', iptrw);
17555: BtnZoomMinus', 'TToolButton', iptrw);
17556: BtnZoomPlus', 'TToolButton', iptrw);
17557: Panel1', 'TPanel', iptrw);
17558: LabelBrettgroesse', 'TLabel', iptrw);
17559: CB1SCList', 'TComboBox', iptrw);
17560: ImageListNormal', 'TImageList', iptrw);
17561: spbtnexplore', 'TSpeedButton', iptrw);
17562: spbtnexample', 'TSpeedButton', iptrw);
17563: spbsaveas', 'TSpeedButton', iptrw);
17564: imglogobox', 'TImage', iptrw);
17565:EnlargeFont1', 'TMenuItem', iptrw);
17566:EnlargeFont2', 'TMenuItem', iptrw);
17567:ShrinkFont1', 'TMenuItem', iptrw);
17568:ThreadDemol', 'TMenuItem', iptrw);
17569:HEXEditor1', 'TMenuItem', iptrw);
17570:HEXView1', 'TMenuItem', iptrw);
17571:HEXInspect1', 'TMenuItem', iptrw);
17572:SynExporterHTML1', 'TSynExporterHTML', iptrw);
17573:ExporttoHTML1', 'TMenuItem', iptrw);
17574:ClassCount1', 'TMenuItem', iptrw);
17575:HTMLOutput1', 'TMenuItem', iptrw);
17576:HEXEditor2', 'TMenuItem', iptrw);
17577:Minesweeper1', 'TMenuItem', iptrw);
17578:N17', 'TMenuItem', iptrw);
17579:PicturePuzzle1', 'TMenuItem', iptrw);
17580:sbvolhelp', 'TSpeedButton', iptrw);
17581:DependencyWalker1', 'TMenuItem', iptrw);
17582:WebScanner1', 'TMenuItem', iptrw);
17583:View1', 'TMenuItem', iptrw);
17584:mnToolbar1', 'TMenuItem', iptrw);
17585:mnStatusbar2', 'TMenuItem', iptrw);
17586:mnConsole2', 'TMenuItem', iptrw);
17587:mnCoolbar2', 'TMenuItem', iptrw);
17588:mnSplitter2', 'TMenuItem', iptrw);
17589:WebServer1', 'TMenuItem', iptrw);
17590:Tutorial17Server1', 'TMenuItem', iptrw);
17591:Tutorial18Arduinol', 'TMenuItem', iptrw);
17592:SynPerlSyn1', 'TSynPerlSyn', iptrw);
17593:PerlSyntax1', 'TMenuItem', iptrw);
17594:SynPythonSyn1', 'TSynPythonSyn', iptrw);
17595:PythonSyntax1', 'TMenuItem', iptrw);
17596:DMathLibrary1', 'TMenuItem', iptrw);
17597:IntfNavigator1', 'TMenuItem', iptrw);
17598:EnlargeFontConsole1', 'TMenuItem', iptrw);
17599:ShrinkFontConsole1', 'TMenuItem', iptrw);
17600:SetInterfaceList1', 'TMenuItem', iptrw);
17601:popintfList', 'TPopupMenu', iptrw);
17602:intfAdd1', 'TMenuItem', iptrw);
17603:intfDelete1', 'TMenuItem', iptrw);
17604:intfRefactor1', 'TMenuItem', iptrw);
17605:Defactor1', 'TMenuItem', iptrw);
17606:Tutorial19COMArduinol', 'TMenuItem', iptrw);
17607:Tutorial20Regex', 'TMenuItem', iptrw);
17608:N18', 'TMenuItem', iptrw);
17609:ManualE1', 'TMenuItem', iptrw);
17610:FullTextFinder1', 'TMenuItem', iptrw);
17611:Move1', 'TMenuItem', iptrw);
17612:FractalDemol', 'TMenuItem', iptrw);
17613:Tutorial21Android1', 'TMenuItem', iptrw);
17614:Tutorial0Function1', 'TMenuItem', iptrw);
17615:SimuLogBox1', 'TMenuItem', iptrw);
17616:OpenExamples1', 'TMenuItem', iptrw);
17617:SynJScriptSyn1', 'TSynJScriptSyn', iptrw);
17618:JavaScriptSyntax1', 'TMenuItem', iptrw);
17619:Halt1', 'TMenuItem', iptrw);
17620:CodeSearch1', 'TMenuItem', iptrw);
17621:SynRubySyn1', 'TSynRubySyn', iptrw);
17622:RubySyntax1', 'TMenuItem', iptrw);
17623:Undol', 'TMenuItem', iptrw);
17624:SynUNIXShellScriptSyn1', 'TSynUNIXShellScriptSyn', iptrw);
17625:LinuxShellScript1', 'TMenuItem', iptrw);
17626:Rename1', 'TMenuItem', iptrw);
17627:spdcodesearch', 'TSpeedButton', iptrw);
17628:Preview1', 'TMenuItem', iptrw);
17629:Tutorial22Services1', 'TMenuItem', iptrw);
17630:Tutorial23RealTime1', 'TMenuItem', iptrw);
17631:Configuration1', 'TMenuItem', iptrw);
17632:MP3Player1', 'TMenuItem', iptrw);
17633:DLLSpy1', 'TMenuItem', iptrw);
17634:SynURI opener1', 'TSynURI opener', iptrw);
17635:SynURISyn1', 'TSynURISyn', iptrw);
17636:URLinksClicks1', 'TMenuItem', iptrw);
17637:EditReplacel', 'TMenuItem', iptrw);
17638:GotoLine1', 'TMenuItem', iptrw);
17639:ActiveLineColor1', 'TMenuItem', iptrw);
17640:ConfigFile1', 'TMenuItem', iptrw);

```



```

17641:      SortIntfList', 'TMenuItem', iptrw);
17642:      Redo1', 'TMenuItem', iptrw);
17643:      Tutorial24CleanCode1', 'TMenuItem', iptrw);
17644:      Tutorial25Configuration1', 'TMenuItem', iptrw);
17645:      IndentSelection1', 'TMenuItem', iptrw);
17646:      UnindentSection1', 'TMenuItem', iptrw);
17647:      SkyStyle1', 'TMenuItem', iptrw);
17648:      N19', 'TMenuItem', iptrw);
17649:      CountWords1', 'TMenuItem', iptrw);
17650:      imbookmarkimages', 'TImageList', iptrw);
17651:      Bookmark11', 'TMenuItem', iptrw);
17652:      N20', 'TMenuItem', iptrw);
17653:      Bookmark21', 'TMenuItem', iptrw);
17654:      Bookmark31', 'TMenuItem', iptrw);
17655:      Bookmark41', 'TMenuItem', iptrw);
17656:      SynMultiSyn1', 'TSynMultiSyn', iptrw);
17657:
17658:      Procedure IFPS3ClassesPlugin1CompImport( Sender : TObject; x : TPSPascalCompiler)
17659:      Procedure IFPS3ClassesPlugin1ExecImport(Sender: TObject; Exec: TPSExec; x:TPSRuntimeClassImporter);
17660:      Procedure PSScriptCompile( Sender : TPSScript)
17661:      Procedure Compile1Click( Sender : TObject)
17662:      Procedure PSScriptExecute( Sender : TPSScript)
17663:      Procedure open1Click( Sender : TObject)
17664:      Procedure Save2Click( Sender : TObject)
17665:      Procedure Savebefore1Click( Sender : TObject)
17666:      Procedure Largefont1Click( Sender : TObject)
17667:      Procedure FormActivate( Sender : TObject)
17668:      Procedure SBytecode1Click( Sender : TObject)
17669:      Procedure FormKeyPress( Sender : TObject; var Key : Char)
17670:      Procedure Saveas3Click( Sender : TObject)
17671:      Procedure Clear1Click( Sender : TObject)
17672:      Procedure Slinenumbers1Click( Sender : TObject)
17673:      Procedure About1Click( Sender : TObject)
17674:      Procedure Search1Click( Sender : TObject)
17675:      Procedure FormCreate( Sender : TObject)
17676:      Procedure Memo1ReplaceText(Sender:TObject;const ASearch,AReplace:String;Line,Column:Integer;
17677:                               var Action : TSynReplaceAction)
17678:      Procedure Memo1StatusChange( Sender : TObject; Changes : TSynStatusChanges)
17679:      Procedure WordWrap1Click( Sender : TObject)
17680:      Procedure SearchNext1Click( Sender : TObject)
17681:      Procedure Replace1Click( Sender : TObject)
17682:      Function PSScriptNeedFile(Sender:TObject;const OrginFileName:String;var FName,Output:String):Bool;
17683:      Procedure ShowInclude1Click( Sender : TObject)
17684:      Procedure Printout1Click( Sender : TObject)
17685:      Procedure mnuPrintFont1Click( Sender : TObject)
17686:      Procedure Include1Click( Sender : TObject)
17687:      Procedure FormDestroy( Sender : TObject)
17688:      Procedure FormClose( Sender : TObject; var Action : TCloseAction)
17689:      Procedure UpdateView1Click( Sender : TObject)
17690:      Procedure CodeCompletionList1Click( Sender : TObject)
17691:      Procedure SaveOutput1Click( Sender : TObject)
17692:      Procedure ExportClipboard1Click( Sender : TObject)
17693:      Procedure Close1Click( Sender : TObject)
17694:      Procedure Manual1Click( Sender : TObject)
17695:      Procedure LoadLastFile1Click( Sender : TObject)
17696:      Procedure Memo1Change( Sender : TObject)
17697:      Procedure Decompile1Click( Sender : TObject)
17698:      Procedure StepInto1Click( Sender : TObject)
17699:      Procedure StepOut1Click( Sender : TObject)
17700:      Procedure Reset1Click( Sender : TObject)
17701:      Procedure cedebugAfterExecute( Sender : TPSScript)
17702:      Procedure cedebugBreakpoint(Sender:TObject; const FileName:String; Position,Row, Col: Cardinal)
17703:      Procedure cedebugCompile( Sender : TPSScript)
17704:      Procedure cedebugExecute( Sender : TPSScript)
17705:      Procedure cedebugIdle( Sender : TObject)
17706:      Procedure cedebugLineInfo( Sender:TObject;const FileName:String; Position, Row, Col : Cardinal)
17707:      Procedure Memo1SpecialLineColors(Sender: TObject; Line:Int; var Special:Boolean;var FG,BG:TColor);
17708:      Procedure BreakPointMenuClick( Sender : TObject)
17709:      Procedure DebugRun1Click( Sender : TObject)
17710:      Procedure tutorial4Click( Sender : TObject)
17711:      Procedure ImportfromClipboard1Click( Sender : TObject)
17712:      Procedure ImportfromClipboard2Click( Sender : TObject)
17713:      Procedure tutorial1Click( Sender : TObject)
17714:      Procedure ShowSpecChars1Click( Sender : TObject)
17715:      Procedure StatusBar1Db1Click( Sender : TObject)
17716:      Procedure PSScriptLine( Sender : TObject)
17717:      Procedure OpenDirectory1Click( Sender : TObject)
17718:      Procedure procMessClick( Sender : TObject)
17719:      Procedure tbtnUseCaseClick( Sender : TObject)
17720:      Procedure EditFont1Click( Sender : TObject)
17721:      Procedure tutorial21Click( Sender : TObject)
17722:      Procedure tutorial31Click( Sender : TObject)
17723:      Procedure HTMLSyntax1Click( Sender : TObject)
17724:      Procedure ShowInterfaces1Click( Sender : TObject)
17725:      Procedure Tutorial5Click( Sender : TObject)
17726:      Procedure ShowLastException1Click( Sender : TObject)
17727:      Procedure PlayMP31Click( Sender : TObject)
17728:      Procedure AllFunctionsList1Click( Sender : TObject)
17729:      Procedure texSyntax1Click( Sender : TObject)

```

```

17730:  Procedure GetEMails1Click( Sender : TObject)
17731:  procedure DelphiSite1Click(Sender: TObject);
17732:  procedure TerminalStyle1Click(Sender: TObject);
17733:  procedure ReadOnly1Click(Sender: TObject);
17734:  procedure ShellStyle1Click(Sender: TObject);
17735:  procedure Console1Click(Sender: TObject); //3.2
17736:  procedure BigScreen1Click(Sender: TObject);
17737:  procedure Tutorial91Click(Sender: TObject);
17738:  procedure SaveScreenshotClick(Sender: TObject);
17739:  procedure Tutorial101Click(Sender: TObject);
17740:  procedure SQLSyntax1Click(Sender: TObject);
17741:  procedure XMLSyntax1Click(Sender: TObject);
17742:  procedure ComponentCount1Click(Sender: TObject);
17743:  procedure NewInstance1Click(Sender: TObject);
17744:  procedure CSyntax1Click(Sender: TObject);
17745:  procedure Tutorial6Click(Sender: TObject);
17746:  procedure New1Click(Sender: TObject);
17747:  procedure AllObjectsList1Click(Sender: TObject);
17748:  procedure LoadBytecode1Click(Sender: TObject);
17749:  procedure CipherFile1Click(Sender: TObject); //V3.5
17750:  procedure NewInstance1Click(Sender: TObject);
17751:  procedure toolbtnTutorialClick(Sender: TObject);
17752:  procedure Memory1Click(Sender: TObject);
17753:  procedure JavaSyntax1Click(Sender: TObject);
17754:  procedure SyntaxCheck1Click(Sender: TObject);
17755:  procedure ScriptExplorer1Click(Sender: TObject);
17756:  procedure FormOutput1Click(Sender: TObject); //V3.6
17757:  procedure GotoEnd1Click(Sender: TObject);
17758:  procedure AllResourceList1Click(Sender: TObject);
17759:  procedure tbtn6resClick(Sender: TObject); //V3.7
17760:  procedure Info1Click(Sender: TObject);
17761:  procedure Tutorial10Statistics1Click(Sender: TObject);
17762:  procedure Tutorial11Forms1Click(Sender: TObject);
17763:  procedure Tutorial12SQL1Click(Sender: TObject); //V3.8
17764:  procedure ResourceExplore1Click(Sender: TObject);
17765:  procedure Info1Click(Sender: TObject);
17766:  procedure CryptoBox1Click(Sender: TObject);
17767:  procedure ModulesCount1Click(Sender: TObject);
17768:  procedure N4GewinntGame1Click(Sender: TObject);
17769:  procedure PHPSyntax1Click(Sender: TObject);
17770:  procedure SerialRS2321Click(Sender: TObject);
17771:  procedure CSyntax2Click(Sender: TObject);
17772:  procedure Calculator1Click(Sender: TObject);
17773:  procedure Tutorial13Ciphering1Click(Sender: TObject);
17774:  procedure Tutorial14Async1Click(Sender: TObject);
17775:  procedure PHPSyntax1Click(Sender: TObject);
17776:  procedure BtnZoomPlusClick(Sender: TObject);
17777:  procedure BtnZoomMinusClick(Sender: TObject);
17778:  procedure btnClassReportClick(Sender: TObject);
17779:  procedure ThreadDemo1Click(Sender: TObject);
17780:  procedure HEXView1Click(Sender: TObject);
17781:  procedure ExporttoHTML1Click(Sender: TObject);
17782:  procedure Minesweeper1Click(Sender: TObject);
17783:  procedure PicturePuzzle1Click(Sender: TObject); //V3.9
17784:  procedure sbvclhelpClick(Sender: TObject);
17785:  procedure DependencyWalker1Click(Sender: TObject);
17786:  procedure CB1SCListDrawItem(Control:TWinControl;Index:Int;aRect:TRect;State:TOwnerDrawState);
17787:  procedure WebScanner1Click(Sender: TObject);
17788:  procedure mnToolbar1Click(Sender: TObject);
17789:  procedure mnStatusbar2Click(Sender: TObject);
17790:  procedure mnConsole2Click(Sender: TObject);
17791:  procedure mnCoolbar2Click(Sender: TObject);
17792:  procedure mnSplitter2Click(Sender: TObject);
17793:  procedure WebServer1Click(Sender: TObject);
17794:  procedure PerlSyntax1Click(Sender: TObject);
17795:  procedure PythonSyntax1Click(Sender: TObject);
17796:  procedure DMathLibrary1Click(Sender: TObject);
17797:  procedure IntfNavigator1Click(Sender: TObject);
17798:  procedure FullTextFinder1Click(Sender: TObject);
17799:  function AppName: string;
17800:  function ScriptName: string;
17801:  function LastName: string;
17802:  procedure FractalDemo1Click(Sender: TObject);
17803:  procedure SimuLogBox1Click(Sender: TObject);
17804:  procedure OpenExamples1Click(Sender: TObject);
17805:  procedure Halt1Click(Sender: TObject);
17806:  procedure Stop;
17807:  procedure CodeSearch1Click(Sender: TObject);
17808:  procedure RubySyntax1Click(Sender: TObject);
17809:  procedure Undo1Click(Sender: TObject);
17810:  procedure LinuxShellScript1Click(Sender: TObject);
17811:  procedure WebScannerDirect(urls: string);
17812:  procedure WebScanner(urls: string);
17813:  procedure LoadInterfaceList2;
17814:  procedure DLLSpy1Click(Sender: TObject);
17815:  procedure Memo1Db1Click(Sender: TObject);
17816:  procedure URILinksClicks1Click(Sender: TObject);
17817:  procedure GotoLine1Click(Sender: TObject);
17818:  procedure ConfigFile1Click(Sender: TObject);

```

```

17819: Procedure SortIntfListClick( Sender : TObject)
17820: Procedure Redo1Click( Sender : TObject)
17821: Procedure Tutorial24CleanCode1Click( Sender : TObject)
17822: Procedure IndentSelection1Click( Sender : TObject)
17823: Procedure UnindentSection1Click( Sender : TObject)
17824: Procedure SkyStyle1Click( Sender : TObject)
17825: Procedure CountWords1Click( Sender : TObject)
17826: Procedure Memo1PlaceBookmark( Sender : TObject; var Mark : TSynEditMark)
17827: Procedure Memo1GutterClick(Sender:TObject;Button:TMouseButton;X,Y,Line:Integer;Mark:TSynEditMark);
17828: Procedure Bookmark11Click( Sender : TObject)
17829: Procedure Bookmark21Click( Sender : TObject)
17830: Procedure Bookmark31Click( Sender : TObject)
17831: Procedure Bookmark41Click( Sender : TObject)
17832: Procedure SynMultiSyn1CustomRange(Sender:TSynMultiSyn;Operation:TRangeOperation;var Range:Pointer);
17833: 'STATMemoryReport', 'boolean', iptrw);
17834: 'IPPort', 'integer', iptrw);
17835: 'COMPort', 'integer', iptrw);
17836: 'lbintfList', 'TListBox', iptrw);
17837: Function GetStatChange : boolean
17838: Procedure SetStatChange( vstat : boolean)
17839: Function GetActFileName : string
17840: Procedure SetActFileName( vname : string)
17841: Function GetLastFileName : string
17842: Procedure SetLastFileName( vname : string)
17843: Procedure WebScannerDirect( urls : string)
17844: Procedure LoadInterfaceList2
17845: Function GetStatExecuteShell : boolean
17846: Procedure DoEditorExecuteCommand( EditorCommand : word)
17847: function GetActiveLineColor: TColor
17848: procedure SetActiveLineColor(acolor: TColor)
17849: procedure ScriptListBox1Click(Sender: TObject);
17850: procedure Memo2KeyPress(Sender: TObject; var Key: Char);
17851: procedure EnlargeGutter1Click(Sender: TObject);
17852: procedure Tetris1Click(Sender: TObject);
17853: procedure ToDoList1Click(Sender: TObject);
17854: procedure ProcessList1Click(Sender: TObject);
17855: procedure MetricReport1Click(Sender: TObject);
17856: procedure ProcessList1Click(Sender: TObject);
17857: procedure TCPSockets1Click(Sender: TObject);
17858: procedure ConfigUpdate1Click(Sender: TObject);
17859: procedure ADOWorkbench1Click(Sender: TObject);
17860: procedure SocketServer1Click(Sender: TObject);
17861: procedure FormDemo1Click(Sender: TObject);
17862: procedure Richedit1Click(Sender: TObject);
17863: procedure SimpleBrowser1Click(Sender: TObject);
17864: procedure DOSShell1Click(Sender: TObject);
17865: procedure SynExport1Click(Sender: TObject);
17866: procedure ExporttoRTF1Click(Sender: TObject);
17867: procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
17868: procedure SOAPTester1Click(Sender: TObject);
17869: procedure Sniffer1Click(Sender: TObject);
17870: procedure AutoDetectSyntax1Click(Sender: TObject);
17871: procedure FPlot1Click(Sender: TObject);
17872: procedure PasStyle1Click(Sender: TObject);
17873: procedure Tutorial183RGBLED1Click(Sender: TObject);
17874: procedure ReversilClick(Sender: TObject);
17875: procedure ManualmaXbox1Click(Sender: TObject);
17876: procedure BlaisePascalMagazine1Click(Sender: TObject);
17877: procedure AddToDo1Click(Sender: TObject);
17878: procedure CreateGUID1Click(Sender: TObject);
17879: procedure Tutorial27XML1Click(Sender: TObject);
17880: procedure CreatedLLStub1Click(Sender: TObject);
17881: procedure Tutorial28DLL1Click(Sender: TObject);
17882: procedure ResetKeyPressed;
17883: procedure FileChanges1Click(Sender: TObject);
17884: procedure OpenGLTry1Click(Sender: TObject);
17885: procedure AllUnitList1Click(Sender: TObject);
17886:
17887:
17888: //-----
17889: //*****mX4 Editor SynEdit Tools API *****
17890: //-----
17891: (*-----*)
17892: procedure SIRegister_TCustomSynEdit(CL: TPSPascalCompiler);
17893: begin
17894:   //with RegClassS(CL,'TCustomControl', 'TCustomSynEdit') do
17895:   with FindClass('TCustomControl'),'TCustomSynEdit') do begin
17896:     Constructor Create( AOwner : TComponent)
17897:     SelStart', 'Integer', iptrw);
17898:     SelEnd', 'Integer', iptrw); AlwaysShowCaret', 'Boolean', iptrw);
17899:     Procedure UpdateCaret
17900:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2:word;SS2:TShiftState);
17901:     Procedure AddKey(Command: TSynEditorCommand;Key1:word;SS1:TShiftState; Key2: word;SS2:TShiftState);
17902:     Procedure BeginUndoBlock
17903:     Procedure BeginUpdate
17904:     Function CaretInView : Boolean
17905:     Function CharIndexToRowCol( Index : integer) : TBufferCoord
17906:     Procedure Clear
17907:     Procedure ClearAll

```

```

17908: Procedure ClearBookMark( BookMark : Integer)
17909: Procedure ClearSelection
17910: Procedure CommandProcessor( Command : TSynEditorCommand; AChar : char; Data : pointer)
17911: Procedure ClearUndo
17912: Procedure CopyToClipboard
17913: Procedure CutToClipboard
17914: Procedure DoCopyToClipboard( const SText : string)
17915: Procedure EndUndoBlock
17916: Procedure EndUpdate
17917: Procedure EnsureCursorPosVisible
17918: Procedure EnsureCursorPosVisibleEx( ForceToMiddle : Boolean)
17919: Procedure FindMatchingBracket
17920: Function GetMatchingBracket : TBufferCoord
17921: Function GetMatchingBracketEx( const APoint : TBufferCoord) : TBufferCoord
17922: Procedure ExecuteCommand( Command : TSynEditorCommand; AChar : char; Data : pointer)
17923: Function GetBookMark( BookMark : integer; var X, Y : integer) : boolean
17924: Function GetHighlighterAttriAtRowCol( const XY : TBufferCoord; var Token : string; var Attri
17925: : TSynHighlighterAttributes) : boolean
17926: Function GetHighlighterAttriAtRowColEx( const XY : TBufferCoord; var Token : string;
17927: var TokenType, Start : Integer; var Attri:TSynHighlighterAttributes):boolean
17928: Function GetPositionOfMouse( out aPos : TBufferCoord) : Boolean
17929: Function GetWordAtRowCol( const XY : TBufferCoord) : string
17930: Procedure GotoBookMark( BookMark : Integer)
17931: Procedure GotoLineAndCenter( ALine : Integer)
17932: Function IdentChars : TSynIdentChars
17933: Procedure InvalidateGutter
17934: Procedure InvalidateGutterLine( aLine : integer)
17935: Procedure InvalidateGutterLines( FirstLine, LastLine : integer)
17936: Procedure InvalidateLine( Line : integer)
17937: Procedure InvalidateLines( FirstLine, LastLine : integer)
17938: Procedure InvalidateSelection
17939: Function IsBookmark( BookMark : integer) : boolean
17940: Function IsPointInSelection( const Value : TBufferCoord) : boolean
17941: Procedure LockUndo
17942: Function BufferToDisplayPos( const p : TBufferCoord) : TDisplayCoord
17943: Function DisplayToBufferPos( const p : TDisplayCoord) : TBufferCoord
17944: Function LineToRow( aLine : integer) : integer
17945: Function RowToLine( aRow : integer) : integer
17946: Function NextWordPos : TBufferCoord
17947: Function NextWordPosEx( const XY : TBufferCoord) : TBufferCoord
17948: Procedure PasteFromClipboard
17949: Function WordStart : TBufferCoord
17950: Function WordStartEx( const XY : TBufferCoord) : TBufferCoord
17951: Function WordEnd : TBufferCoord
17952: Function WordEndEx( const XY : TBufferCoord) : TBufferCoord
17953: Function PrevWordPos : TBufferCoord
17954: Function PrevWordPosEx( const XY : TBufferCoord) : TBufferCoord
17955: Function PixelsToRowColumn( aX, aY : integer) : TDisplayCoord
17956: Function PixelsToNearestRowColumn( aX, aY : integer) : TDisplayCoord
17957: Procedure Redo
17958: Procedure RegisterCommandHandler(const AHandlerProc:THookedCommandEvent;AHandlerData:pointer));
17959: Function RowColumnToPixels( const RowCol : TDisplayCoord) : TPoint
17960: Function RowColToCharIndex( RowCol : TBufferCoord) : integer
17961: Function SearchReplace( const ASearch, AReplace:string; AOptions:TSynSearchOptions): integer
17962: Procedure SelectAll
17963: Procedure SetBookMark( BookMark : Integer; X : Integer; Y : Integer)
17964: Procedure SetCaretAndSelection( const ptCaret, ptBefore, ptAfter : TBufferCoord)
17965: Procedure SetDefaultKeystrokes
17966: Procedure SetSelWord
17967: Procedure SetWordBlock( Value : TBufferCoord)
17968: Procedure Undo
17969: Procedure UnlockUndo
17970: Procedure UnregisterCommandHandler( AHandlerProc : THookedCommandEvent)
17971: Procedure AddKeyUpHandler( aHandler : TKeyEvent)
17972: Procedure RemoveKeyUpHandler( aHandler : TKeyEvent)
17973: Procedure AddKeyDownHandler( aHandler : TKeyEvent)
17974: Procedure RemoveKeyDownHandler( aHandler : TKeyEvent)
17975: Procedure AddKeyPressHandler( aHandler : TKeyPressEvent)
17976: Procedure RemoveKeyPressHandler( aHandler : TKeyPressEvent)
17977: Procedure AddFocusControl( aControl : TWinControl)
17978: Procedure RemoveFocusControl( aControl : TWinControl)
17979: Procedure AddMouseDownHandler( aHandler : TMouseEvent)
17980: Procedure RemoveMouseDownHandler( aHandler : TMouseEvent)
17981: Procedure AddMouseUpHandler( aHandler : TMouseEvent)
17982: Procedure RemoveMouseUpHandler( aHandler : TMouseEvent)
17983: Procedure AddMouseCursorHandler( aHandler : TMouseEvent)
17984: Procedure RemoveMouseCursorHandler( aHandler : TMouseEvent)
17985: Procedure SetLinesPointer( ASynEdit : TCustomSynEdit)
17986: Procedure RemoveLinesPointer
17987: Procedure HookTextBuffer( aBuffer : TSynEditStringList; aUndo, aRedo : TSynEditUndoList)
17988: Procedure UnHookTextBuffer
17989: BlockBegin, 'TBufferCoord', iptrw);
17990: BlockEnd, 'TBufferCoord', iptrw);
17991: CanPaste, 'Boolean', iptr);
17992: CanRedo, 'boolean', iptr);
17993: CanUndo, 'boolean', iptr);
17994: CaretX, 'Integer', iptrw);
17995: CaretY, 'Integer', iptrw);
17996: CaretXY, 'TBufferCoord', iptrw);

```



```

17997:   ActiveLineColor', 'TColor', iptrw);
17998:   DisplayX', 'Integer', iptr);
17999:   DisplayY', 'Integer', iptr);
18000:   DisplayXY', 'TDisplayCoord', iptr);
18001:   DisplayLineCount', 'integer', iptr);
18002:   CharsInWindow', 'Integer', iptr);
18003:   CharWidth', 'integer', iptr);
18004:   Font', 'TFont', iptrw);
18005:   GutterWidth', 'Integer', iptr);
18006:   Highlighter', 'TSynCustomHighlighter', iptrw);
18007:   LeftChar', 'Integer', iptrw);
18008:   LineHeight', 'integer', iptr);
18009:   LinesInWindow', 'Integer', iptr);
18010:   LineText', 'string', iptrw);
18011:   Lines', 'TStrings', iptrw);
18012:   Marks', 'TSynEditMarkList', iptr);
18013:   MaxScrollWidth', 'integer', iptrw);
18014:   Modified', 'Boolean', iptrw);
18015:   PaintLock', 'Integer', iptr);
18016:   ReadOnly', 'Boolean', iptrw);
18017:   SearchEngine', 'TSynEditSearchCustom', iptrw);
18018:   SelAvail', 'Boolean', iptr); SelLength', 'integer', iptrw);
18019:   SelTabBlock', 'Boolean', iptr);
18020:   SelTabLine', 'Boolean', iptr);
18021:   SelText', 'string', iptrw);
18022:   StateFlags', 'TSynStateFlags', iptr);
18023:   Text', 'string', iptrw);
18024:   TopLine', 'Integer', iptrw);
18025:   WordAtCursor', 'string', iptr);
18026:   WordAtMouse', 'string', iptr);
18027:   UndoList', 'TSynEditUndoList', iptr);
18028:   RedoList', 'TSynEditUndoList', iptr);
18029:   OnProcessCommand', 'TProcessCommandEvent', iptrw);
18030:   BookMarkOptions', 'TSynBookMarkOpt', iptrw);
18031:   BorderStyle', 'TSynBorderStyle', iptrw);
18032:   ExtraLineSpacing', 'integer', iptrw);
18033:   Gutter', 'TSynGutter', iptrw);
18034:   HideSelection', 'boolean', iptrw);
18035:   InsertCaret', 'TSynEditCaretType', iptrw);
18036:   InsertMode', 'boolean', iptrw);
18037:   IsScrolling', 'Boolean', iptr);
18038:   Keystrokes', 'TSynEditKeyStrokes', iptrw);
18039:   MaxUndo', 'Integer', iptrw);
18040:   Options', 'TSynEditorOptions', iptrw);
18041:   OverwriteCaret', 'TSynEditCaretType', iptrw);
18042:   RightEdge', 'Integer', iptrw); RightEdgeColor', 'TColor', iptrw);
18043:   ScrollHintColor', 'TColor', iptrw);
18044:   ScrollHintFormat', 'TScrollHintFormat', iptrw);
18045:   ScrollBars', 'TScrollStyle', iptrw);
18046:   SelectedColor', 'TSynSelectedColor', iptrw);
18047:   SelectionMode', 'TSynSelectionMode', iptrw);
18048:   ActiveSelectionMode', 'TSynSelectionMode', iptrw);
18049:   TabWidth', 'integer', iptrw); WantReturns', 'boolean', iptrw);
18050:   WantTabs', 'boolean', iptrw); WordWrap', 'boolean', iptrw);
18051:   WordWrapGlyph', 'TSynGlyph', iptrw);
18052:   OnChange', 'TNotifyEvent', iptrw);
18053:   OnClearBookmark', 'TPlaceMarkEvent', iptrw);
18054:   OnCommandProcessed', 'TProcessCommandEvent', iptrw);
18055:   OnContextHelp', 'TContextHelpEvent', iptrw);
18056:   OnDropFiles', 'TDropFilesEvent', iptrw);
18057:   OnGutterClick', 'TGutterClickEvent', iptrw);
18058:   OnGutterGetText', 'TGutterGetTextEvent', iptrw);
18059:   OnGutterPaint', 'TGutterPaintEvent', iptrw);
18060:   OnMouseCursor', 'TMouseCursorEvent', iptrw);
18061:   OnPaint', 'TPaintEvent', iptrw);
18062:   OnPlaceBookmark', 'TPlaceMarkEvent', iptrw);
18063:   OnProcessUserCommand', 'TProcessCommandEvent', iptrw);
18064:   OnReplaceText', 'TReplaceTextEvent', iptrw);
18065:   OnSpecialLineColors', 'TSpecialLineColorsEvent', iptrw);
18066:   OnStatusChange', 'TStatusChangeEvent', iptrw);
18067:   OnPaintTransient', 'TPaintTransient', iptrw);
18068:   OnScroll', 'TScrollEvent', iptrw);
18069:   end;
18070: end;
18071: Procedure RegisterPlaceableHighlighter(highlighter : TSynCustomHighlighterClass)
18072: Function GetPlaceableHighlighters : TSynHighlighterList
18073: Function EditorCommandToDescrString( Cmd : TSynEditorCommand) : string
18074: Function EditorCommandToCodeString( Cmd : TSynEditorCommand) : string
18075: Procedure GetEditorCommandValues( Proc : TGetStrProc)
18076: Procedure GetEditorCommandExtended( Proc : TGetStrProc)
18077: Function IdentToEditorCommand( const Ident : string; var Cmd : longint) : boolean
18078: Function EditorCommandToIdent( Cmd : longint; var Ident : string) : boolean
18079: Function ConvertCodeStringToExtended( AString : String) : String
18080: Function ConvertExtendedToCodeString( AString : String) : String
18081: Function ConvertExtendedToCommand( AString : String) : TSynEditorCommand
18082: Function ConvertCodeStringToCommand( AString : String) : TSynEditorCommand
18083: Function IndexToEditorCommand( const AIndex : Integer) : Integer
18084:
18085:   TSynEditorOption = (

```

```

18086:    eoAltSetsColumnMode,      //Holding down the Alt Key will put the selection mode into columnar format
18087:    eoAutoIndent,            //Will indent caret on newlines with same amount of leading whitespace as
18088:                             // preceding line
18089:    eoAutoSizeMaxScrollWidth, //Automatically resizes the MaxScrollWidth property when inserting text
18090:    eoDisableScrollArrows,    //Disables the scroll bar arrow buttons when you can't scroll in that
18091:                             //direction any more
18092:    eoDragDropEditing,        //Allows to select a block of text and drag it within document to another
18093:                             // location
18094:    eoDropFiles,              //Allows the editor accept OLE file drops
18095:    eoEnhanceHomeKey,         //enhances home key positioning, similar to visual studio
18096:    eoEnhanceEndKey,          //enhances End key positioning, similar to JDeveloper
18097:    eoGroupUndo,              //When undoing/redoin actions, handle all continous changes the same kind
18098:                             // in one call
18099:                             //instead undoing/redoin each command separately
18100:    eoHalfPageScroll,         //When scrolling with page-up and page-down commands, only scroll a half
18101:                             //page at a time
18102:    eoHideShowScrollbars,     //if enabled, then scrollbars will only show if necessary.
18103:    If you have ScrollPastEOL, then it the horizontal bar will always be there (it uses MaxLength instead)
18104:    eoKeepCaretX,             //When moving through lines w/o cursor Past EOL, keeps X position of cursor
18105:    eoNoCaret,                //Makes it so the caret is never visible
18106:    eoNoSelection,            //Disables selecting text
18107:    eoRightMouseMovesCursor,  //When clicking with right mouse for popup menu, moves cursor to location
18108:    eoScrollByOneLess,        //Forces scrolling to be one less
18109:    eoScrollHintFollows,      //The scroll hint follows the mouse when scrolling vertically
18110:    eoScrollPastEof,          //Allows the cursor to go past the end of file marker
18111:    eoScrollPastEol,          //Allows cursor to go past last character into white space at end of a line
18112:    eoShowScrollHint,         //Shows a hint of the visible line numbers when scrolling vertically
18113:    eoShowSpecialChars,       //Shows the special Characters
18114:    eoSmartTabDelete,         //similar to Smart Tabs, but when you delete characters
18115:    eoSmartTabs,              //When tabbing, cursor will go to non-white space character of previous line
18116:    eoSpecialLineDefaultFg,    //disables the foreground text color override using OnSpecialLineColor event
18117:    eoTabIndent,              //If active <Tab>and<Shift><Tab> act block indent,unindent when text select
18118:    eoTabsToSpaces,           //Converts a tab character to a specified number of space characters
18119:    eoTrimTrailingSpaces      //Spaces at the end of lines will be trimmed and not saved
18120:
18121:    *****Important Editor Short Cuts*****);
18122:    Double click to select a word and count words with highlightning.
18123:    Triple click to select a line.
18124:    CTRL+SHIFT+click to extend a selection.
18125:    Drag with the ALT key down to select columns of text !!!
18126:    Drag and drop is supported.
18127:    Type CTRL+Z to undo and SHIFT+CTRL+Z to redo.
18128:    Type CTRL+A to select all.
18129:    Type CTRL+N to set a new line.
18130:    Type CTRL+T to delete a line or token. //Tokenizer
18131:    Type CTRL+C to copy to clipboard. Type CTRL+V to paste from clipboard.
18132:    Type CTRL+Shift+T to add ToDo in line and list.
18133:    Type CTRL+Shift+[0..9] to set bookmarks. //Bookmark
18134:    Type CTRL[0..9] to jump or get to bookmarks.
18135:    Type Home to position cursor at beginning of current line and End to position it at end of line.
18136:    Type CTRL+Home to position cursor at start of doc and CTRL+End to position it at end of document.
18137:    Page Up and Page Down work as expected.
18138:    CTRL+Page Up sends cursor to top of viewed portion and CTRL+Page Down sends it to bottom.
18139:    using http://pp4s.co.uk/main/tu-form2-help-demo-laz.html
18140:
18141:    {$ Short Key Positions Ctrl<A-Z>: }
18142:    def
18143:    <A> Select All
18144:    <B> Count Words
18145:    <C> Copy
18146:    <D> Internet Start
18147:    <E> Script List
18148:    <F> Find
18149:    <G> Goto
18150:    <H> Mark Line
18151:    <I> Interface List
18152:    <J> Code Completion
18153:    <K> Console
18154:    <L> Interface List Box
18155:    <M> Font Larger -
18156:    <N> New Line
18157:    <O> Open File
18158:    <P> Font Smaller +
18159:    <Q> Quit
18160:    <R> Replace
18161:    <S> Save!
18162:    <T> Delete Line
18163:    <U> Use Case Editor
18164:    <V> Paste
18165:    <W> URI Links
18166:    <X> Reserved for coding use internal
18167:    <Y> Delete Line
18168:    <Z> Undo
18169:
18170:    ref
18171:    F1 Help
18172:    F2 Syntax Check
18173:    F3 Search Next
18174:    F4 New Instance

```

PC08 E:\maxbox3\mXGit39988\maxbox3\docs\maxbox_extract_funclist399.txt
<http://www.softwareschule.ch/maxbox.htm>

```

18264: - Add on IP Protocol Sniffer with List View
18265: - Add on OpenGL mX Robot Demo for android
18266:
18267: - Menu: Help/Tools as a Tool Section with DOS Opener
18268: - Menu Editor: export the code as RTF File
18269: - Menu: Help/Tools/Syn Export your code as available in HTML or RTF
18270: - Menu: Help/Tools/ with <Alt H-S-D> you start the DOS Shell
18271: - Context: Auto Detect of Syntax depending on file extension
18272: - Code: some Windows API function start with w in the name like wGetAtomName();
18273: - IDE Close - if you cant close the box then reset it <Ctrl F2> in menu /Debug
18274: - IDE File Check with menu ..View/File Changes/...
18275:
18276: - using DLL example in maxbox: //function: {*****}
18277: Function GetProcessMemoryInfo(Process: THandle; var MemoryCounters: TProcessMemoryCounters;
18278:                               cb: DWORD): BOOL; //stdcall;
18279: External 'GetProcessMemoryInfo@psapi.dll stdcall';
18280:
18281: Function OpenProcess(dwDesiredAccess:DWORD; bInheritHandle:BOOL; dwProcessId: DWORD):THandle;
18282: External 'OpenProcess@kernel32.dll stdcall';
18283:
18284: PCT Precompile Technology , mX4 ScriptStudio
18285: Indy, JCL, Jedi, VCL, Systools, TurboPower, Fundamentals, ExtendedRTL, Synedit
18286: DMath, devC, Graphics32, ExtPascal, mX4, LCL, CLX, FCL, CPort and more
18287: emax layers: system-package-component-unit-class-function-block
18288: new keywords def ref using maxCalcF
18289: UML: use case act class state seq pac comp dep - lib lab
18290: FBX Lib, psAPI, SMS Cell Module, OpenGL, Borland Tools
18291: Tutorials, 30 Units add, VCL constructors, controls plus, unit list
18292: 2 Tutorials, 36 Units add, Synapse V40, LDAP, OpenSSL, AVScan
18293: https://unibe-ch.academia.edu/MaxKleiner
18294: www.slideshare.net/maxkleiner1
18295: http://www.scribd.com/max_kleiner
18296:
18297:
18298: *****
18299: unit List asm internal end
18300: *****
18301: 01 unit RRegister_StrUtils_Routines(exec); //Delphi
18302: 02 unit SRegister_IdStrings //Indy Sockets
18303: 03 unit RRegister_niSTRING_Routines(Exec); //from RegEx
18304: 04 unit uPSI_fMain Functions; //maxbox Open Tools API
18305: 05 unit IFSI_WinFormlpuzzle; //maxbox
18306: 06 unit RRegister_LinarBitmap_Routines(Exec); //ImageFileLibBCB
18307: 07 unit RegisterDateTimeLibrary_R(exec); //Delphi
18308: 08 unit RRegister_MathMax_Routines(exec); //Jedi & Delphi
18309: 09 unit RRegister_IdGlobal_Routines(exec); //Indy Sockets
18310: 10 unit RRegister_SysUtils_Routines(Exec); //Delphi
18311: 11 unit uPSI_IdTCPConnection; //Indy some functions
18312: 12 unit uPSCCompiler.pas; //PS kernel functions
18313: 13 unit uPSI_DBCommon; //DB Common_Routines and Types
18314: 14 unit uPSI_Printers.pas //Delphi VCL
18315: 15 unit uPSI_MPlayer.pas //Delphi VCL
18316: 16 unit uPSC_comobj; //COM Functions
18317: 17 unit uPSI_Clipbrd; //Delphi VCL
18318: 18 unit Filectrl in IFSI_SysUtils_max; //VCL Runtime
18319: 19 unit uPSI_SqlExpr; //DBX3
18320: 20 unit uPSI_ADODB; //ADODB
18321: 21 unit uPSI_StrHlpr; //String Helper Routines
18322: 22 unit uPSI_DateUtils; //Expansion to DateTimeLib
18323: 23 unit uPSI_FileUtils; //Expansion to Sys/File Utils
18324: 24 unit JUtils / gsUtils; //Jedi / Metabase
18325: 25 unit JvFunctions_max; //Jedi Functions
18326: 26 unit HTTPParser; //Delphi VCL
18327: 27 unit HTTPUtil; //Delphi VCL
18328: 28 unit uPSI_XMLUtil; //Delphi VCL
18329: 29 unit uPSI_SOAPHTTPClient; //Delphi VCL SOAP WebService V3.5
18330: 30 unit uPSI_Contrns; //Delphi RTL Container of Classes
18331: 31 unit uPSI_MaskUtils; //RTL Edit and Mask functions
18332: 32 unit uPSI_MyBigInt; //big integer class with Math
18333: 33 unit uPSI_ConvUtils; //Delphi VCL Conversions engine
18334: 34 unit Types_Variants; //Delphi\Win32\rtl\sys
18335: 35 unit uPSI_IdHashSHA1; //Indy Crypto Lib
18336: 36 unit uPSI_IdHashMessageDigest //Indy Crypto;
18337: 37 unit uPSI_IdASN1Util; //Indy ASN1Utility Routines;
18338: 38 unit uPSI_IdLogFile; //Indy Logger from LogBase
18339: 39 unit uPSI_IdIcmpClient; //Indy Ping ICMP
18340: 40 unit uPSI_IdHashMessageDigest_max //Indy Crypto &OpenSSL;
18341: 41 unit uPSI_FileCtrl; //Delphi RTL
18342: 42 unit uPSI_Outline; //Delphi VCL
18343: 43 unit uPSI_ScktComp; //Delphi RTL
18344: 44 unit uPSI_Calendar; //Delphi VCL
18345: 45 unit uPSI_VListView //VListView;
18346: 46 unit uPSI_DBGrids; //Delphi VCL
18347: 47 unit uPSI_DBCtrls; //Delphi VCL
18348: 48 unit ide_debugoutput; //maxbox
18349: 49 unit uPSI_ComCtrls; //Delphi VCL
18350: 50 unit uPSC_stdctrls+; //Delphi VCL
18351: 51 unit uPSI_Dialogs; //Delphi VCL
18352: 52 unit uPSI_StdConvs; //Delphi RTL

```



```

18353: 53 unit uPSI_DBClient; //Delphi RTL
18354: 54 unit uPSI_DBPlatform; //Delphi RTL
18355: 55 unit uPSI_Provider; //Delphi RTL
18356: 56 unit uPSI_FMTBcd; //Delphi RTL
18357: 57 unit uPSI_DBCGrids; //Delphi VCL
18358: 58 unit uPSI_CDSUtil; //MIDAS
18359: 59 unit uPSI_VarHlpr; //Delphi RTL
18360: 60 unit uPSI_ExtDlgs; //Delphi VCL
18361: 61 unit sdpStopwatch; //maXbox
18362: 62 unit uPSI_JclStatistics; //JCL
18363: 63 unit uPSI_JclLogic; //JCL
18364: 64 unit uPSI_JclMiscel; //JCL
18365: 65 unit uPSI_JclMath_max; //JCL RTL
18366: 66 unit uPSI_uTPLb_StreamUtils; //LockBox 3
18367: 67 unit uPSI_MathUtils; //BCB
18368: 68 unit uPSI_JclMultimedia; //JCL
18369: 69 unit uPSI_WideStrUtils; //Delphi API/RTL
18370: 70 unit uPSI_GraphUtil; //Delphi RTL
18371: 71 unit uPSI_TypeTrans; //Delphi RTL
18372: 72 unit uPSI_HTTPApp; //Delphi VCL
18373: 73 unit uPSI_DBWeb; //Delphi VCL
18374: 74 unit uPSI_DBBdeWeb; //Delphi VCL
18375: 75 unit uPSI_DBXpressWeb; //Delphi VCL
18376: 76 unit uPSI_ShadowWnd; //Delphi VCL
18377: 77 unit uPSI_ToolWin; //Delphi VCL
18378: 78 unit uPSI_Tabs; //Delphi VCL
18379: 79 unit uPSI_JclGraphUtils; //JCL
18380: 80 unit uPSI_JclCounter; //JCL
18381: 81 unit uPSI_JclSysInfo; //JCL
18382: 82 unit uPSI_JclSecurity; //JCL
18383: 83 unit uPSI_JclFileUtils; //JCL
18384: 84 unit uPSI_IdUserAccounts; //Indy
18385: 85 unit uPSI_IdAuthentication; //Indy
18386: 86 unit uPSI_uTPLb_AES; //LockBox 3
18387: 87 unit uPSI_IdHashSHA1; //LockBox 3
18388: 88 unit uTPLb_BlockCipher; //LockBox 3
18389: 89 unit uPSI_ValEdit.pas; //Delphi VCL
18390: 90 unit uPSI_JvVCLUtils; //JCL
18391: 91 unit uPSI_JvDBUtil; //JCL
18392: 92 unit uPSI_JvDBUtils; //JCL
18393: 93 unit uPSI_JvAppUtils; //JCL
18394: 94 unit uPSI_JvCtrlUtils; //JCL
18395: 95 unit uPSI_JvFormToHtml; //JCL
18396: 96 unit uPSI_JvParsing; //JCL
18397: 97 unit uPSI_SerDlgs; //Toolbox
18398: 98 unit uPSI_Serial; //Toolbox
18399: 99 unit uPSI_JvComponent; //JCL
18400: 100 unit uPSI_JvCalc; //JCL
18401: 101 unit uPSI_JvBdeUtils; //JCL
18402: 102 unit uPSI_JvDateUtil; //JCL
18403: 103 unit uPSI_JvGenetic; //JCL
18404: 104 unit uPSI_JclBase; //JCL
18405: 105 unit uPSI_JvUtils; //JCL
18406: 106 unit uPSI_JvStrUtil; //JCL
18407: 107 unit uPSI_JvStrUtils; //JCL
18408: 108 unit uPSI_JvFileUtil; //JCL
18409: 109 unit uPSI_JvMemoryInfos; //JCL
18410: 110 unit uPSI_JvComputerInfo; //JCL
18411: 111 unit uPSI_JvgCommClasses; //JCL
18412: 112 unit uPSI_JvgLogics; //JCL
18413: 113 unit uPSI_JvLED; //JCL
18414: 114 unit uPSI_JvTurtle; //JCL
18415: 115 unit uPSI_SortThds; unit uPSI_ThSort; //maXbox
18416: 116 unit uPSI_JvgUtils; //JCL
18417: 117 unit uPSI_JvExprParser; //JCL
18418: 118 unit uPSI_HexDump; //Borland
18419: 119 unit uPSI_DBLogDlg; //VCL
18420: 120 unit uPSI_SqlTimSt; //RTL
18421: 121 unit uPSI_JvHtmlParser; //JCL
18422: 122 unit uPSI_JvgXMLSerializer; //JCL
18423: 123 unit uPSI_JvJCLUtils; //JCL
18424: 124 unit uPSI_JvStrings; //JCL
18425: 125 unit uPSI_uTPLb_IntegerUtils; //TurboPower
18426: 126 unit uPSI_uTPLb_HugeCardinal; //TurboPower
18427: 127 unit uPSI_uTPLb_HugeCardinalUtils; //TurboPower
18428: 128 unit uPSI_SynRegExpr; //SynEdit
18429: 129 unit uPSI_StUtils; //SysTools4
18430: 130 unit uPSI_StToHTML; //SysTools4
18431: 131 unit uPSI_StStrms; //SysTools4
18432: 132 unit uPSI_StFIN; //SysTools4
18433: 133 unit uPSI_StAstroP; //SysTools4
18434: 134 unit uPSI_StStat; //SysTools4
18435: 135 unit uPSI_StNetCon; //SysTools4
18436: 136 unit uPSI_StDecMth; //SysTools4
18437: 137 unit uPSI_StOStr; //SysTools4
18438: 138 unit uPSI_StPtrns; //SysTools4
18439: 139 unit uPSI_StNetMessage; //SysTools4
18440: 140 unit uPSI_StMath; //SysTools4
18441: 141 unit uPSI_StExpEng; //SysTools4

```

```

18442: 142 unit uPSI_StCRC; //SysTools4
18443: 143 unit uPSI_StExport, //SysTools4
18444: 144 unit uPSI_StExpLog, //SysTools4
18445: 145 unit uPSI_ActnList; //Delphi VCL
18446: 146 unit uPSI_jpeg; //Borland
18447: 147 unit uPSI_StRandom; //SysTools4
18448: 148 unit uPSI_StDict; //SysTools4
18449: 149 unit uPSI_StBCD; //SysTools4
18450: 150 unit uPSI_StTxtDat; //SysTools4
18451: 151 unit uPSI_StRegEx; //SysTools4
18452: 152 unit uPSI_IMouse; //VCL
18453: 153 unit uPSI_SyncObjs; //VCL
18454: 154 unit uPSI_AsyncCalls; //Hausladen
18455: 155 unit uPSI_ParallelJobs; //Saraiva
18456: 156 unit uPSI_Variants; //VCL
18457: 157 unit uPSI_VarCmplx; //VCL Wolfram
18458: 158 unit uPSI_DTDSchema; //VCL
18459: 159 unit uPSI_ShLwApi; //Brakel
18460: 160 unit uPSI_IBUtils; //VCL
18461: 161 unit uPSI_CheckLst; //VCL
18462: 162 unit uPSI_JvSimpleXml; //JCL
18463: 163 unit uPSI_JclSimpleXml; //JCL
18464: 164 unit uPSI_JvXmlDatabase; //JCL
18465: 165 unit uPSI_JvMaxPixel; //JCL
18466: 166 unit uPSI_JvItemsSearchs; //JCL
18467: 167 unit uPSI_StExpEng2; //SysTools4
18468: 168 unit uPSI_StGenLog; //SysTools4
18469: 169 unit uPSI_JvLogFile; //Jcl
18470: 170 unit uPSI_CPort; //ComPort Lib v4.11
18471: 171 unit uPSI_CPortCtl; //ComPort
18472: 172 unit uPSI_CPortEsc; //ComPort
18473: 173 unit BarCodeScanner; //ComPort
18474: 174 unit uPSI_JvGraph; //JCL
18475: 175 unit uPSI_JvComCtrls; //JCL
18476: 176 unit uPSI_GUITesting; //D Unit
18477: 177 unit uPSI_JvFindFiles; //JCL
18478: 178 unit uPSI_StSystem; //SysTools4
18479: 179 unit uPSI_JvKeyboardStates; //JCL
18480: 180 unit uPSI_JvMail; //JCL
18481: 181 unit uPSI_JclConsole; //JCL
18482: 182 unit uPSI_JclLANMan; //JCL
18483: 183 unit uPSI_IdCustomHTTPServer; //Indy
18484: 184 unit IdHTTPServer; //Indy
18485: 185 unit uPSI_IdTCPSTServer; //Indy
18486: 186 unit uPSI_IdSocketHandle; //Indy
18487: 187 unit uPSI_IdIOHandlerSocket; //Indy
18488: 188 unit IdIOHandler; //Indy
18489: 189 unit uPSI_cutils; //Bloodshed
18490: 190 unit uPSI_BoldUtils; //boldsoft
18491: 191 unit uPSI_IdSimpleServer; //Indy
18492: 192 unit uPSI_IdSSLOpenSSL; //Indy
18493: 193 unit uPSI_IdMultipartFormData; //Indy
18494: 194 unit uPSI_SynURIOpener; //SynEdit
18495: 195 unit uPSI_PerlRegEx; //PCRE
18496: 196 unit uPSI_IdHeaderList; //Indy
18497: 197 unit uPSI_StFirst; //SysTools4
18498: 198 unit uPSI_JvCtrls; //JCL
18499: 199 unit uPSI_IdTrivialFTPBase; //Indy
18500: 200 unit uPSI_IdTrivialFTP; //Indy
18501: 201 unit uPSI_IdUDPBBase; //Indy
18502: 202 unit uPSI_IdUDPCClient; //Indy
18503: 203 unit uPSI_otypes; //for DMATH.DLL
18504: 204 unit uPSI_ShellAPI; //Borland
18505: 205 unit uPSI_IdRemoteCMDClient; //Indy
18506: 206 unit uPSI_IdRemoteCMDServer; //Indy
18507: 207 unit IdRexecServer; //Indy
18508: 208 unit IdRexec; (unit uPSI_IdRexec;) //Indy
18509: 209 unit IdUDPServer; //Indy
18510: 210 unit IdTimeUDPServer; //Indy
18511: 211 unit IdTimeServer; //Indy
18512: 212 unit IdTimeUDP; (unit uPSI_IdUDPServer;) //Indy
18513: 213 unit uPSI_IdIPWatch; //Indy
18514: 214 unit uPSI_IdIrcServer; //Indy
18515: 215 unit uPSI_IdMessageCollection; //Indy
18516: 216 unit uPSI_cPEM; //Fundamentals 4
18517: 217 unit uPSI_cFundamentUtils; //Fundamentals 4
18518: 218 unit uPSI_uwinplot; //DMATH
18519: 219 unit uPSI_xrtl_util_CPUUtils; //ExtendedRTL
18520: 220 unit uPSI_GR32_System; //Graphics32
18521: 221 unit uPSI_cFileUtils; //Fundamentals 4
18522: 222 unit uPSI_cDateTime; (timemachine) //Fundamentals 4
18523: 223 unit uPSI_cTimers; (high precision timer) //Fundamentals 4
18524: 224 unit uPSI_cRandom; //Fundamentals 4
18525: 225 unit uPSI_ueval; //DMATH
18526: 226 unit uPSI_xrtl_net_URIUtils; //ExtendedRTL
18527: 227 unit xrtl_net_URIUtils; //ExtendedRTL
18528: 228 unit uPSI_ufft; (FFT) //DMATH
18529: 229 unit uPSI_DBXChannel; //Delphi
18530: 230 unit uPSI_DBXIndyChannel; //Delphi Indy

```

```

18531: 231 unit uPSI_xrtl_util_COMCat; //ExtendedRTL
18532: 232 unit uPSI_xrtl_util_StrUtils; //ExtendedRTL
18533: 233 unit uPSI_xrtl_util_VariantUtils; //ExtendedRTL
18534: 234 unit uPSI_xrtl_util_FileUtils; //ExtendedRTL
18535: 235 unit xrtl_util_Compat; //ExtendedRTL
18536: 236 unit uPSI_OleAuto; //Borland
18537: 237 unit uPSI_xrtl_util_COMUtils; //ExtendedRTL
18538: 238 unit uPSI_CmAdmCtl; //Borland
18539: 239 unit uPSI_ValEdit2; //VCL
18540: 240 unit uPSI_GR32; //Graphics32
18541: 241 unit uPSI_GR32_Image; //Graphics32
18542: 242 unit uPSI_xrtl_util_TimeUtils; //ExtendedRTL
18543: 243 unit uPSI_xrtl_util_TimeZone; //ExtendedRTL
18544: 244 unit uPSI_xrtl_util_TimeStamp; //ExtendedRTL
18545: 245 unit uPSI_xrtl_util_Map; //ExtendedRTL
18546: 246 unit uPSI_xrtl_util_Set; //ExtendedRTL
18547: 247 unit uPSI_CPortMonitor; //ComPort
18548: 248 unit uPSI_StIniStm; //SysTools4
18549: 249 unit uPSI_GR32_ExtImage; //Graphics32
18550: 250 unit uPSI_GR32_OrdinalMaps; //Graphics32
18551: 251 unit uPSI_GR32_Rasterizers; //Graphics32
18552: 252 unit uPSI_xrtl_util_Exception; //ExtendedRTL
18553: 253 unit uPSI_xrtl_util_Value; //ExtendedRTL
18554: 254 unit uPSI_xrtl_util_Compare; //ExtendedRTL
18555: 255 unit uPSI_FlatSB; //VCL
18556: 256 unit uPSI_JvAnalogClock; //JCL
18557: 257 unit uPSI_JvAlarms; //JCL
18558: 258 unit uPSI_JvSQLS; //JCL
18559: 259 unit uPSI_JvDBSecur; //JCL
18560: 260 unit uPSI_JvDBQBE; //JCL
18561: 261 unit uPSI_JvStarfield; //JCL
18562: 262 unit uPSI_JvCLMiscal; //JCL
18563: 263 unit uPSI_JvProfiler32; //JCL
18564: 264 unit uPSI_JvDirectories; //JCL
18565: 265 unit uPSI_JclSchedule; //JCL
18566: 266 unit uPSI_JclSvcCtrl; //JCL
18567: 267 unit uPSI_JvSoundControl; //JCL
18568: 268 unit uPSI_JvBDESQLScript; //JCL
18569: 269 unit uPSI_JvgDigits; //JCL>
18570: 270 unit uPSI_ImgList; //TCustomImageList
18571: 271 unit uPSI_JclMIDI; //JCL>
18572: 272 unit uPSI_JclWinMidi; //JCL>
18573: 273 unit uPSI_JclNTFS; //JCL>
18574: 274 unit uPSI_JclAppInst; //JCL>
18575: 275 unit uPSI_JvRle; //JCL>
18576: 276 unit uPSI_JvRas32; //JCL>
18577: 277 unit uPSI_JvImageDrawThread; //JCL>
18578: 278 unit uPSI_JvImageWindow; //JCL>
18579: 279 unit uPSI_JvTransparentForm; //JCL>
18580: 280 unit uPSI_JvWinDialogs; //JCL>
18581: 281 unit uPSI_JvSimLogic; //JCL>
18582: 282 unit uPSI_JvSimIndicator; //JCL>
18583: 283 unit uPSI_JvSimPID; //JCL>
18584: 284 unit uPSI_JvSimPIDLinker; //JCL>
18585: 285 unit uPSI_IdRFCReply; //Indy
18586: 286 unit uPSI_IdIdent; //Indy
18587: 287 unit uPSI_IdIdentServer; //Indy
18588: 288 unit uPSI_JvPatchFile; //JCL
18589: 289 unit uPSI_StNetPfm; //SysTools4
18590: 290 unit uPSI_StNet; //SysTools4
18591: 291 unit uPSI_JclPeImage; //JCL
18592: 292 unit uPSI_JclPrint; //JCL
18593: 293 unit uPSI_JclMime; //JCL
18594: 294 unit uPSI_JvRichEdit; //JCL
18595: 295 unit uPSI_JvDBRichEd; //JCL
18596: 296 unit uPSI_JvDice; //JCL
18597: 297 unit uPSI_JvFloatEdit; //JCL 3.9.8
18598: 298 unit uPSI_JvDirFrm; //JCL
18599: 299 unit uPSI_JvDualList; //JCL
18600: 300 unit uPSI_JvSwitch; ///JCL
18601: 301 unit uPSI_JvTimerLst; ///JCL
18602: 302 unit uPSI_JvMemTable; //JCL
18603: 303 unit uPSI_JvObjStr; //JCL
18604: 304 unit uPSI_StLArr; //SysTools4
18605: 305 unit uPSI_StWmDCpy; //SysTools4
18606: 306 unit uPSI_StText; //SysTools4
18607: 307 unit uPSI_StNTLog; //SysTools4
18608: 308 unit uPSI_xrtl_math_Integer; //ExtendedRTL
18609: 309 unit uPSI_JvImagPrvw; //JCL
18610: 310 unit uPSI_JvFormPatch; //JCL
18611: 311 unit uPSI_JvPicClip; //JCL
18612: 312 unit uPSI_JvDataConv; //JCL
18613: 313 unit uPSI_JvCpuUsage; //JCL
18614: 314 unit uPSI_JclUnitConv_mX2; //JCL
18615: 315 unit JvDualListForm; //JCL
18616: 316 unit uPSI_JvCpuUsage2; //JCL
18617: 317 unit uPSI_JvParserForm; //JCL
18618: 318 unit uPSI_JvJanTreeView; //JCL
18619: 319 unit uPSI_JvTransLED; //JCL

```

```

18620: 320 unit uPSI_JvPlaylist; //JCL
18621: 321 unit uPSI_JvFormAutoSize; //JCL
18622: 322 unit uPSI_JvYearGridEditForm; //JCL
18623: 323 unit uPSI_JvMarkupCommon; //JCL
18624: 324 unit uPSI_JvChart; //JCL
18625: 325 unit uPSI_JvXPCore; //JCL
18626: 326 unit uPSI_JvXPCoreUtils; //JCL
18627: 327 unit uPSI_StatsClasses; //mX4
18628: 328 unit uPSI_ExtCtrls2; //VCL
18629: 329 unit uPSI_JvUrlGrabbers; //JCL
18630: 330 unit uPSI_JvXmlTree; //JCL
18631: 331 unit uPSI_JvWavePlayer; //JCL
18632: 332 unit uPSI_JvUnicodeCanvas; //JCL
18633: 333 unit uPSI_JvTFUtils; //JCL
18634: 334 unit uPSI_IdServerIOHandler; //Indy
18635: 335 unit uPSI_IdServerIOHandlerSocket; //Indy
18636: 336 unit uPSI_IdMessageCoder; //Indy
18637: 337 unit uPSI_IdMessageCoderMIME; //Indy
18638: 338 unit uPSI_IdMIMETypes; //Indy
18639: 339 unit uPSI_JvConverter; //JCL
18640: 340 unit uPSI_JvCsvParse; //JCL
18641: 341 unit uPSI_umath; unit uPSI_ugamma; //DMath
18642: 342 unit uPSI_ExcelExport; (Nat:TJsExcelExport) //JCL
18643: 343 unit uPSI_JvDBGridExport; //JCL
18644: 344 unit uPSI_JvgExport; //JCL
18645: 345 unit uPSI_JvSerialMaker; //JCL
18646: 346 unit uPSI_JvWin32; //JCL
18647: 347 unit uPSI_JvPaintFX; //JCL
18648: 348 unit uPSI_JvOracleDataSet; (beta) //JCL
18649: 349 unit uPSI_JvValidators; (preview) //JCL
18650: 350 unit uPSI_JvNTEventLog; //JCL
18651: 351 unit uPSI_ShellZipTool; //mX4
18652: 352 unit uPSI_JvJoystick; //JCL
18653: 353 unit uPSI_JvMailSlots; //JCL
18654: 354 unit uPSI_JclComplex; //JCL
18655: 355 unit uPSI_SynPdf; //Synopsis
18656: 356 unit uPSI_Registry; //VCL
18657: 357 unit uPSI_TlHelp32; //VCL
18658: 358 unit uPSI_JclRegistry; //JCL
18659: 359 unit uPSI_JvAirBrush; //JCL
18660: 360 unit uPSI_mORMotReport; //Synopsis
18661: 361 unit uPSI_JclLocales; //JCL
18662: 362 unit uPSI_SynEdit; //SynEdit
18663: 363 unit uPSI_SynEditTypes; //SynEdit
18664: 364 unit uPSI_SynMacroRecorder; //SynEdit
18665: 365 unit uPSI_LongIntList; //SynEdit
18666: 366 unit uPSI_devcutils; //DevC
18667: 367 unit uPSI_SynEditMiscClasses; //SynEdit
18668: 368 unit uPSI_SynEditRegexSearch; //SynEdit
18669: 369 unit uPSI_SynEditHighlighter; //SynEdit
18670: 370 unit uPSI_SynHighlighterPas; //SynEdit
18671: 371 unit uPSI_JvSearchFiles; //JCL
18672: 372 unit uPSI_SynHighlighterAny; //Lazarus
18673: 373 unit uPSI_SynEditKeyCmds; //SynEdit
18674: 374 unit uPSI_SynEditMiscProcs; //SynEdit
18675: 375 unit uPSI_SynEditKbdHandler; //SynEdit
18676: 376 unit uPSI_JvAppInst; //JCL
18677: 377 unit uPSI_JvAppEvent; //JCL
18678: 378 unit uPSI_JvAppCommand; //JCL
18679: 379 unit uPSI_JvAnimTitle; //JCL
18680: 380 unit uPSI_JvAnimatedImage; //JCL
18681: 381 unit uPSI_SynEditExport; //SynEdit
18682: 382 unit uPSI_SynExportHTML; //SynEdit
18683: 383 unit uPSI_SynExportRTF; //SynEdit
18684: 384 unit uPSI_SynEditSearch; //SynEdit
18685: 385 unit uPSI_fMain_back; //maXbox;
18686: 386 unit uPSI_JvZoom; //JCL
18687: 387 unit uPSI_PMrand; //PM
18688: 388 unit uPSI_JvSticker; //JCL
18689: 389 unit uPSI_XmlVerySimple; //mX4
18690: 390 unit uPSI_Services; //ExtPascal
18691: 391 unit uPSI_ExtPascalUtils; //ExtPascal
18692: 392 unit uPSI_SocketsDelphi; //ExtPascal
18693: 393 unit uPSI_StBarC; //SysTools
18694: 394 unit uPSI_StDbBarC; //SysTools
18695: 395 unit uPSI_StBarPN; //SysTools
18696: 396 unit uPSI_StDbPNBC; //SysTools
18697: 397 unit uPSI_StDb2DBC; //SysTools
18698: 398 unit uPSI_StMoney; //SysTools
18699: 399 unit uPSI_JvForth; //JCL
18700: 400 unit uPSI_RestRequest; //mX4
18701: 401 unit uPSI_HttpRESTConnectionIndy; //mX4
18702: 402 unit uPSI_JvXmlDatabase; //update //JCL
18703: 403 unit uPSI_StAstro; //SysTools
18704: 404 unit uPSI_StSort; //SysTools
18705: 405 unit uPSI_StDate; //SysTools
18706: 406 unit uPSI_StDateSt; //SysTools
18707: 407 unit uPSI_StBase; //SysTools
18708: 408 unit uPSI_StVInfo; //SysTools

```



```

18709: 409 unit uPSI_JvBrowseFolder; //JCL
18710: 410 unit uPSI_JvBoxProcs; //JCL
18711: 411 unit uPSI_urandom; (unit uranuvag;) //DMath
18712: 412 unit uPSI_usimann; (unit ugenalg;) //DMath
18713: 413 unit uPSI_JvHighlighter; //JCL
18714: 414 unit uPSI_Diff; //mX4
18715: 415 unit uPSI_SpringWinAPI; //DSpring
18716: 416 unit uPSI_StBits; //SysTools
18717: 417 unit uPSI_TomDBQueue; //mX4
18718: 418 unit uPSI_MultilangTranslator; //mX4
18719: 419 unit uPSI_HyperLabel; //mX4
18720: 420 unit uPSI_Starter; //mX4
18721: 421 unit uPSI_FileAssocs; //devC
18722: 422 unit uPSI_devFileMonitorX; //devC
18723: 423 unit uPSI_devrun; //devC
18724: 424 unit uPSI_devExec; //devC
18725: 425 unit uPSI_oysUtils; //devC
18726: 426 unit uPSI_DosCommand; //devC
18727: 427 unit uPSI_CppTokenizer; //devC
18728: 428 unit uPSI_JvHLPParser; //devC
18729: 429 unit uPSI_JclMapi; //JCL
18730: 430 unit uPSI_JclShell; //JCL
18731: 431 unit uPSI_JclCOM; //JCL
18732: 432 unit uPSI_GR32_Math; //Graphics32
18733: 433 unit uPSI_GR32_LowLevel; //Graphics32
18734: 434 unit uPSI_SimpleHl; //mX4
18735: 435 unit uPSI_GR32_Filters; //Graphics32
18736: 436 unit uPSI_GR32_VectorMaps; //Graphics32
18737: 437 unit uPSI_cXMLFunctions; //Fundamentals 4
18738: 438 unit uPSI_JvTimer; //JCL
18739: 439 unit uPSI_cHTTPUtils; //Fundamentals 4
18740: 440 unit uPSI_cTLUtils; //Fundamentals 4
18741: 441 unit uPSI_JclGraphics; //JCL
18742: 442 unit uPSI_JclSynch; //JCL
18743: 443 unit uPSI_IdTelnet; //Indy
18744: 444 unit uPSI_IdTelnetServer; //Indy
18745: 445 unit uPSI_IdEcho; //Indy
18746: 446 unit uPSI_IdEchoServer; //Indy
18747: 447 unit uPSI_IdEchoUDP; //Indy
18748: 448 unit uPSI_IdEchoUDPServer; //Indy
18749: 449 unit uPSI_IdSocks; //Indy
18750: 450 unit uPSI_IdAntiFreezeBase; //Indy
18751: 451 unit uPSI_IdHostnameServer; //Indy
18752: 452 unit uPSI_IdTunnelCommon; //Indy
18753: 453 unit uPSI_IdTunnelMaster; //Indy
18754: 454 unit uPSI_IdTunnelSlave; //Indy
18755: 455 unit uPSI_IdRSH; //Indy
18756: 456 unit uPSI_IdRSHServer; //Indy
18757: 457 unit uPSI_Spring_Cryptography_Utils; //Spring4Delphi
18758: 458 unit uPSI_MapReader; //devC
18759: 459 unit uPSI_LibTar; //devC
18760: 460 unit uPSI_IdStack; //Indy
18761: 461 unit uPSI_IdBlockCipherIntercept; //Indy
18762: 462 unit uPSI_IdChargenServer; //Indy
18763: 463 unit uPSI_IdFTPServer; //Indy
18764: 464 unit uPSI_IdException; //Indy
18765: 465 unit uPSI_utexplot; //DMath
18766: 466 unit uPSI_uwinstr; //DMath
18767: 467 unit uPSI_VarRecUtils; //devC
18768: 468 unit uPSI_JvStringListToHtml; //JCL
18769: 469 unit uPSI_JvStringHolder; //JCL
18770: 470 unit uPSI_IdCoder; //Indy
18771: 471 unit uPSI_SynHighlighterDfm; //Synedit
18772: 472 unit uHighlighterProcs; in 471 //Synedit
18773: 473 unit uPSI_LazFileUtils; //LCL
18774: 474 unit uPSI_IDECmdLine; //LCL
18775: 475 unit uPSI_lazMasks; //LCL
18776: 476 unit uPSI_ip_misc; //mX4
18777: 477 unit uPSI_Barcode; //LCL
18778: 478 unit uPSI_SimpleXML; //LCL
18779: 479 unit uPSI_JclIniFiles; //JCL
18780: 480 unit uPSI_D2XXUnit; {$X-} //FTDI
18781: 481 unit uPSI_JclDateTime; //JCL
18782: 482 unit uPSI_JclEDI; //JCL
18783: 483 unit uPSI_JclMiscel2; //JCL
18784: 484 unit uPSI_JclValidation; //JCL
18785: 485 unit uPSI_JclAnsiStrings; {-PString} //JCL
18786: 486 unit uPSI_SynEditMiscProcs2; //Synedit
18787: 487 unit uPSI_JclStreams; //JCL
18788: 488 unit uPSI_QRCode; //mX4
18789: 489 unit uPSI_BlockSocket; //ExtPascal
18790: 490 unit uPSI_Masks,Utils //VCL
18791: 491 unit uPSI_synauttil; //Synapse!
18792: 492 unit uPSI_JclMath_Class; //JCL RTL
18793: 493 unit ugamdist; //Gamma function //DMath
18794: 494 unit uibeta, ucorrel; //IBeta //DMath
18795: 495 unit uPSI_SRMgr; //mX4
18796: 496 unit uPSI_HotLog; //mX4
18797: 497 unit uPSI_DebugBox; //mX4

```

```

18798: 498 unit uPSI_ustrings; //DMath
18799: 499 unit uPSI_uregtest; //DMath
18800: 500 unit uPSI_usimplex; //DMath
18801: 501 unit uPSI_uhyper; //DMath
18802: 502 unit uPSI_IdHLL7; //Indy
18803: 503 unit uPSI_IdIPMCastBase; //Indy
18804: 504 unit uPSI_IdIPMCastServer; //Indy
18805: 505 unit uPSI_IdIPMCastClient; //Indy
18806: 506 unit uPSI_unlfit; //nlregression //DMath
18807: 507 unit uPSI_IdRawHeaders; //Indy
18808: 508 unit uPSI_IdRawClient; //Indy
18809: 509 unit uPSI_IdRawFunctions; //Indy
18810: 510 unit uPSI_IdTCPStream; //Indy
18811: 511 unit uPSI_IdSNPP; //Indy
18812: 512 unit uPSI_St2DBarC; //SysTools
18813: 513 unit uPSI_ImageWin; //FTL //VCL
18814: 514 unit uPSI_CustomDrawTreeView; //FTL //VCL
18815: 515 unit uPSI_GraphWin; //FTL //VCL
18816: 516 unit uPSI_actionMain; //FTL //VCL
18817: 517 unit uPSI_StSpawn; //SysTools
18818: 518 unit uPSI_CtlPanel; //VCL
18819: 519 unit uPSI_IdLPR; //Indy
18820: 520 unit uPSI_SockRequestInterpreter; //Indy
18821: 521 unit uPSI_ulambert; //DMath
18822: 522 unit uPSI_ucholesk; //DMath
18823: 523 unit uPSI_SimpleDS; //VCL
18824: 524 unit uPSI_DBXSqlScanner; //VCL
18825: 525 unit uPSI_DBXMetaDataUtil; //VCL
18826: 526 unit uPSI_Chart; //TEE
18827: 527 unit uPSI_TeeProcs; //TEE
18828: 528 unit mXBDEUtils; //mX4
18829: 529 unit uPSI_MDIEdit; //VCL
18830: 530 unit uPSI_CopyPrsr; //VCL
18831: 531 unit uPSI_SockApp; //VCL
18832: 532 unit uPSI_AppEvnts; //VCL
18833: 533 unit uPSI_ExtActns; //VCL
18834: 534 unit uPSI_TeEngine; //TEE
18835: 535 unit uPSI_CoolMain; //browser //VCL
18836: 536 unit uPSI_StCRC; //SysTools
18837: 537 unit uPSI_StDecMth2; //SysTools
18838: 538 unit uPSI_frmExportMain; //Synedit
18839: 539 unit uPSI_SynDBEdit; //Synedit
18840: 540 unit uPSI_SynEditWildcardSearch; //Synedit
18841: 541 unit uPSI_BoldComUtils; //BOLD
18842: 542 unit uPSI_BoldIsoDateTime; //BOLD
18843: 543 unit uPSI_BoldGUIDUtils; //inCOMUtils //BOLD
18844: 544 unit uPSI_BoldXMLRequests; //BOLD
18845: 545 unit uPSI_BoldStringList; //BOLD
18846: 546 unit uPSI_BoldFileHandler; //BOLD
18847: 547 unit uPSI_BoldContainers; //BOLD
18848: 548 unit uPSI_BoldQueryUserDlg; //BOLD
18849: 549 unit uPSI_BoldWinINET; //BOLD
18850: 550 unit uPSI_BoldQueue; //BOLD
18851: 551 unit uPSI_JvPcx; //JCL
18852: 552 unit uPSI_IdWhois; //Indy
18853: 553 unit uPSI_IdWhoIsServer; //Indy
18854: 554 unit uPSI_IdGopher; //Indy
18855: 555 unit uPSI_IdDateTimeStamp; //Indy
18856: 556 unit uPSI_IdDayTimeServer; //Indy
18857: 557 unit uPSI_IdDayTimeUDP; //Indy
18858: 558 unit uPSI_IdDayTimeUDPServer; //Indy
18859: 559 unit uPSI_IdDICTServer; //Indy
18860: 560 unit uPSI_IdDiscardServer; //Indy
18861: 561 unit uPSI_IdDiscardUDPServer; //Indy
18862: 562 unit uPSI_IdMappedFTP; //Indy
18863: 563 unit uPSI_IdMappedPortTCP; //Indy
18864: 564 unit uPSI_IdGopherServer; //Indy
18865: 565 unit uPSI_IdQotdServer; //Indy
18866: 566 unit uPSI_JvRgbToHtml; //JCL
18867: 567 unit uPSI_JvRemLog; //JCL
18868: 568 unit uPSI_JvSysComp; //JCL
18869: 569 unit uPSI_JvTMTL; //JCL
18870: 570 unit uPSI_JvWinampAPI; //JCL
18871: 571 unit uPSI_MSysUtils; //mX4
18872: 572 unit uPSI_ESBMaths; //ESB
18873: 573 unit uPSI_ESBMaths2; //ESB
18874: 574 unit uPSI_uLkJSON; //Lk
18875: 575 unit uPSI_ZURL; //Zeos
18876: 576 unit uPSI_ZSysUtils; //Zeos
18877: 577 unit unaUtils internals; //UNA
18878: 578 unit uPSI_ZMatchPattern; //Zeos
18879: 579 unit uPSI_ZClasses; //Zeos
18880: 580 unit uPSI_ZCollections; //Zeos
18881: 581 unit uPSI_ZEncoding; //Zeos
18882: 582 unit uPSI_IdRawBase; //Indy
18883: 583 unit uPSI_IdNTLM; //Indy
18884: 584 unit uPSI_IdNNTP; //Indy
18885: 585 unit uPSI_usniffer; //PortScanForm //mX4
18886: 586 unit uPSI_IdCoderMIME; //Indy

```

```

18887: 587 unit uPSI_IdCoderUUE; //Indy
18888: 588 unit uPSI_IdCoderXxE; //Indy
18889: 589 unit uPSI_IdCoder3to4; //Indy
18890: 590 unit uPSI_IdCookie; //Indy
18891: 591 unit uPSI_IdCookieManager; //Indy
18892: 592 unit uPSI_WDosSocketUtils; //WDos
18893: 593 unit uPSI_WDosPlcUtils; //WDos
18894: 594 unit uPSI_WDosPorts; //WDos
18895: 595 unit uPSI_WDosResolvers; //WDos
18896: 596 unit uPSI_WDosTimers; //WDos
18897: 597 unit uPSI_WDosPlcs; //WDos
18898: 598 unit uPSI_WDosPneumatics; //WDos
18899: 599 unit uPSI_IdFingerServer; //Indy
18900: 600 unit uPSI_IdDNSResolver; //Indy
18901: 601 unit uPSI_IdHTTPWebBrokerBridge; //Indy
18902: 602 unit uPSI_IdIntercept; //Indy
18903: 603 unit uPSI_IdIPMCastBase; //Indy
18904: 604 unit uPSI_IdLogBase; //Indy
18905: 605 unit uPSI_IdIOHandlerStream; //Indy
18906: 606 unit uPSI_IdMappedPortUDP; //Indy
18907: 607 unit uPSI_IdQOTDUDPServer; //Indy
18908: 608 unit uPSI_IdQOTDUDP; //Indy
18909: 609 unit uPSI_IdSysLog; //Indy
18910: 610 unit uPSI_IdSysLogServer; //Indy
18911: 611 unit uPSI_IdSysLogMessage; //Indy
18912: 612 unit uPSI_IdTimeServer; //Indy
18913: 613 unit uPSI_IdTimeUDP; //Indy
18914: 614 unit uPSI_IdTimeUDPServer; //Indy
18915: 615 unit uPSI_IdUserAccounts; //Indy
18916: 616 unit uPSI_TextUtils; //mX4
18917: 617 unit uPSI_MandelbrotEngine; //mX4
18918: 618 unit uPSI_delphi_arduino_Unit1; //mX4
18919: 619 unit uPSI_DTDSchema2; //mX4
18920: 620 unit uPSI_fplotMain; //DMath
18921: 621 unit uPSI_FindFileIter; //mX4
18922: 622 unit uPSI_PppState; (JclStrHashMap) //PPP
18923: 623 unit uPSI_PppParser; //PPP
18924: 624 unit uPSI_PppLexer; //PPP
18925: 625 unit uPSI_PCharUtils; //PPP
18926: 626 unit uPSI_uJSON; //WU
18927: 627 unit uPSI_JclStrHashMap; //JCL
18928: 628 unit uPSI_JclHookExcept; //JCL
18929: 629 unit uPSI_EncdDecd; //VCL
18930: 630 unit uPSI_SockAppReg; //VCL
18931: 631 unit uPSI_PJFileHandle; //PJ
18932: 632 unit uPSI_PJEnvVars; //PJ
18933: 633 unit uPSI_PJPipe; //PJ
18934: 634 unit uPSI_PJPipeFilters; //PJ
18935: 635 unit uPSI_PJConsoleApp; //PJ
18936: 636 unit uPSI_UConsoleAppEx; //PJ
18937: 637 unit uPSI_DbxSocketChannelNative; //VCL
18938: 638 unit uPSI_DbxDataGenerator; //VCL
18939: 639 unit uPSI_DBXClient; //VCL
18940: 640 unit uPSI_IdLogEvent; //Indy
18941: 641 unit uPSI_Reversi; //mX4
18942: 642 unit uPSI_Geometry; //mX4
18943: 643 unit uPSI_IdSMTPServer; //Indy
18944: 644 unit uPSI_Textures; //mX4
18945: 645 unit uPSI_IBX; //VCL
18946: 646 unit uPSI_IWDBCommon; //VCL
18947: 647 unit uPSI_SortGrid; //mX4
18948: 648 unit uPSI_IB; //VCL
18949: 649 unit uPSI_IBScript; //VCL
18950: 650 unit uPSI_JvCSVBaseControls; //JCL
18951: 651 unit uPSI_Jvg3DColors; //JCL
18952: 652 unit uPSI_JvHLEditor; //beat //JCL
18953: 653 unit uPSI_JvShellHook; //JCL
18954: 654 unit uPSI_DBCommon2; //VCL
18955: 655 unit uPSI_JvSHFileOperation; //JCL
18956: 656 unit uPSI_uFileExport; //mX4
18957: 657 unit uPSI_JvDialogs; //JCL
18958: 658 unit uPSI_JvDBTreeView; //JCL
18959: 659 unit uPSI_JvDBUltimGrid; //JCL
18960: 660 unit uPSI_JvDBQueryParamsForm; //JCL
18961: 661 unit uPSI_JvExControls; //JCL
18962: 662 unit uPSI_JvBDEMemTable; //JCL
18963: 663 unit uPSI_JvCommStatus; //JCL
18964: 664 unit uPSI_JvMailSlots2; //JCL
18965: 665 unit uPSI_JvgWinMask; //JCL
18966: 666 unit uPSI_StEclipse; //SysTools
18967: 667 unit uPSI_StMime; //SysTools
18968: 668 unit uPSI_StList; //SysTools
18969: 669 unit uPSI_StMerge; //SysTools
18970: 670 unit uPSI_StStrS; //SysTools
18971: 671 unit uPSI_StTree; //SysTools
18972: 672 unit uPSI_StVArr; //SysTools
18973: 673 unit uPSI_StRegIni; //SysTools
18974: 674 unit uPSI_urkf; //DMath
18975: 675 unit uPSI_usvd; //DMath

```

```

18976: 676 unit uPSI_DepWalkUtils; //JCL
18977: 677 unit uPSI_OptionsFrm; //JCL
18978: 678 unit yuvconverts; //mX4
18979: 679 uPSI_JvPropAutoSave; //JCL
18980: 680 uPSI_AcAPI; //alcinoe
18981: 681 uPSI_AviCap; //alcinoe
18982: 682 uPSI_ALAVLBinaryTree; //alcinoe
18983: 683 uPSI_ALFcnMisc; //alcinoe
18984: 684 uPSI_ALStringList; //alcinoe
18985: 685 uPSI_ALQuickSortList; //alcinoe
18986: 686 uPSI_ALStaticText; //alcinoe
18987: 687 uPSI_ALJSONDoc; //alcinoe
18988: 688 uPSI_ALGSMComm; //alcinoe
18989: 689 uPSI_ALWindows; //alcinoe
18990: 690 uPSI_ALMultiPartFormDataParser; //alcinoe
18991: 691 uPSI_ALHttpCommon; //alcinoe
18992: 692 uPSI_ALWebSpider; //alcinoe
18993: 693 uPSI_ALHttpClient; //alcinoe
18994: 694 uPSI_ALFcnHTML; //alcinoe
18995: 695 uPSI_ALFTPClient; //alcinoe
18996: 696 uPSI_ALInternetMessageCommon; //alcinoe
18997: 697 uPSI_ALWininetHttpClient; //alcinoe
18998: 698 uPSI_ALWinInetFTPClient; //alcinoe
18999: 699 uPSI_ALWinHttpWrapper; //alcinoe
19000: 700 uPSI_ALWinHttpClient; //alcinoe
19001: 701 uPSI_ALFcnWinSock; //alcinoe
19002: 702 uPSI_ALFcnSQL; //alcinoe
19003: 703 uPSI_ALFcnCGI; //alcinoe
19004: 704 uPSI_ALFcnExecute; //alcinoe
19005: 705 uPSI_ALFcnFile; //alcinoe
19006: 706 uPSI_ALFcnMime; //alcinoe
19007: 707 uPSI_ALPhpRunner; //alcinoe
19008: 708 uPSI_ALGraphic; //alcinoe
19009: 709 uPSI_ALIniFiles; //alcinoe
19010: 710 uPSI_ALMemCachedClient; //alcinoe
19011: 711 unit uPSI_MyGrids; //mX4
19012: 712 uPSI_ALMultiPartMixedParser; //alcinoe
19013: 713 uPSI_ALSMTPClient; //alcinoe
19014: 714 uPSI_ALNNTTPClient; //alcinoe
19015: 715 uPSI_ALHintBalloon; //alcinoe
19016: 716 unit uPSI_ALXmlDoc; //alcinoe
19017: 717 unit uPSI_IPCThrd; //VCL
19018: 718 unit uPSI_MonForm; //VCL
19019: 719 unit uPSI_TeCanvas; //Orpheus
19020: 720 unit uPSI_Ovcmisc; //Orpheus
19021: 721 unit uPSI_ovcfile; //Orpheus
19022: 722 unit uPSI_ovcstate; //Orpheus
19023: 723 unit uPSI_ovccoco; //Orpheus
19024: 724 unit uPSI_ovcrvexp; //Orpheus
19025: 725 unit uPSI_OvcFormatSettings; //Orpheus
19026: 726 unit uPSI_OvcUtils; //Orpheus
19027: 727 unit uPSI_ovcstore; //Orpheus
19028: 728 unit uPSI_ovcstr; //Orpheus
19029: 729 unit uPSI_ovcmru; //Orpheus
19030: 730 unit uPSI_ovccmd; //Orpheus
19031: 731 unit uPSI_ovctimer; //Orpheus
19032: 732 unit uPSI_ovcintl; //Orpheus
19033: 733 uPSI_AfCircularBuffer; //AsyncFree
19034: 734 uPSI_AfUtils; //AsyncFree
19035: 735 uPSI_AfSafeSync; //AsyncFree
19036: 736 uPSI_AfComPortCore; //AsyncFree
19037: 737 uPSI_AfComPort; //AsyncFree
19038: 738 uPSI_AfPortControls; //AsyncFree
19039: 739 uPSI_AfDataDispatcher; //AsyncFree
19040: 740 uPSI_AfViewers; //AsyncFree
19041: 741 uPSI_AfDataTerminal; //AsyncFree
19042: 742 uPSI_SimplePortMain; //AsyncFree
19043: 743 unit uPSI_ovcclock; //Orpheus
19044: 744 unit uPSI_o32intl; //Orpheus
19045: 745 unit uPSI_o32ledlabel; //Orpheus
19046: 746 unit uPSI_ALMySQLClient; //alcinoe
19047: 747 unit uPSI_ALFBXClient; //alcinoe
19048: 748 unit uPSI_ALFcnSQL; //alcinoe
19049: 749 unit uPSI_AsyncTimer; //mX4
19050: 750 unit uPSI_ApplicationFileIO; //mX4
19051: 751 unit uPSI_PsAPI; //VCL6
19052: 752 uPSI_ovcuser; //Orpheus
19053: 753 uPSI_ovcurl; //Orpheus
19054: 754 uPSI_ovcvlb; //Orpheus
19055: 755 uPSI_ovccolor; //Orpheus
19056: 756 uPSI_ALFBXLib; //alcinoe
19057: 757 uPSI_ovcmeter; //Orpheus
19058: 758 uPSI_ovcpeakm; //Orpheus
19059: 759 uPSI_O32BGSty; //Orpheus
19060: 760 uPSI_ovcBidi; //Orpheus
19061: 761 uPSI_ovctcary; //Orpheus
19062: 762 uPSI_DXPUtils; //mX4
19063: 763 uPSI_ALMultiPartBaseParser; //alcinoe
19064: 764 uPSI_ALMultiPartAlternativeParser; //alcinoe

```



```

19065: 765 uPSI_ALPOP3Client; //alcinoe
19066: 766 uPSI_SmallUtils; //mX4
19067: 767 uPSI_MakeApp; //mX4
19068: 768 uPSI_O32MouseMon; //Orpheus
19069: 769 uPSI_OvcCache; //Orpheus
19070: 770 uPSI_ovccalc; //Orpheus
19071: 771 uPSI_Joystick; //OpenGL
19072: 772 uPSI_ScreenSaver; //OpenGL
19073: 773 uPSI_XCollection; //OpenGL
19074: 774 uPSI_Polynomials; //OpenGL
19075: 775 uPSI_PersistentClasses, //9.86 //OpenGL
19076: 776 uPSI_VectorLists; //OpenGL
19077: 777 uPSI_XOpenGL; //OpenGL
19078: 778 uPSI_MeshUtils; //OpenGL
19079: 779 unit uPSI_JclSysUtils; //JCL
19080: 780 unit uPSI_JclBorlandTools; //JCL
19081: 781 unit JclFileUtils_max; //JCL
19082: 782 uPSI_AfDataControls; //AsyncFree
19083: 783 uPSI_GLSilhouette; //OpenGL
19084: 784 uPSI_JclSysUtils_class; //JCL
19085: 785 uPSI_JclFileUtils_class; //JCL
19086: 786 uPSI_FileUtil; //JCL
19087: 787 uPSI_changefind; //mX4
19088: 788 uPSI_cmdIntf; //mX4
19089: 789 uPSI_fservise; //mX4
19090: 790 uPSI_Keyboard; //OpenGL
19091: 791 uPSI_VRMLParser; //OpenGL
19092: 792 uPSI_GLFileVRML; //OpenGL
19093: 793 uPSI_Octree; //OpenGL
19094: 794 uPSI_GLPolyhedron; //OpenGL
19095: 795 uPSI_GLCrossPlatform; //OpenGL
19096: 796 uPSI_GLParticles; //OpenGL
19097: 797 uPSI_GLNavigator; //OpenGL
19098: 798 uPSI_GLStarRecord; //OpenGL
19099: 799 uPSI_GLTextureCombiners; //OpenGL
19100: 800 uPSI_GLCanvas; //OpenGL
19101: 801 uPSI_GeometryBB; //OpenGL
19102: 802 uPSI_GeometryCoordinates; //OpenGL
19103: 803 uPSI_VectorGeometry; //OpenGL
19104: 804 uPSI_BumpMapping; //OpenGL
19105: 805 uPSI_TGA; //OpenGL
19106: 806 uPSI_GLVectorFileObjects; //OpenGL
19107: 807 uPSI_IMM; //VCL
19108: 808 uPSI_CategoryButtons; //VCL
19109: 809 uPSI_ButtonGroup; //VCL
19110: 810 uPSI_DbExcept; //VCL
19111: 811 uPSI_AxCtrls; //VCL
19112: 812 uPSI_GL_actorUnit1; //OpenGL
19113: 813 uPSI_StdVCL; //VCL
19114: 814 unit CurvesAndSurfaces; //OpenGL
19115: 815 uPSI_DataAwareMain; //AsyncFree
19116: 816 uPSI_TabNotBk; //VCL
19117: 817 uPSI_udwsfiler; //mX4
19118: 818 uPSI_synaip; //Synapse!
19119: 819 uPSI_synacode; //Synapse
19120: 820 uPSI_synachar; //Synapse
19121: 821 uPSI_synamisc; //Synapse
19122: 822 uPSI_synaser; //Synapse
19123: 823 uPSI_synaicnv; //Synapse
19124: 824 uPSI_tlntsend; //Synapse
19125: 825 uPSI_pingsend; //Synapse
19126: 826 uPSI_blksock; //Synapse
19127: 827 uPSI_asnlutil; //Synapse
19128: 828 uPSI_dnssend; //Synapse
19129: 829 uPSI_clamsend; //Synapse
19130: 830 uPSI_ldapsend; //Synapse
19131: 831 uPSI_mimemess; //Synapse
19132: 832 uPSI_slogsend; //Synapse
19133: 833 uPSI_mimepart; //Synapse
19134: 834 uPSI_mimeinln; //Synapse
19135: 835 uPSI_ftpsend; //Synapse
19136: 836 uPSI_ftptsend; //Synapse
19137: 837 uPSI_htptsend; //Synapse
19138: 838 uPSI_sntpsend; //Synapse
19139: 839 uPSI_smtptsend; //Synapse
19140: 840 uPSI_snmpsend; //Synapse
19141: 841 uPSI_imapsend; //Synapse
19142: 842 uPSI_pop3send; //Synapse
19143: 843 uPSI_nntpsend; //Synapse
19144: 844 uPSI_ssl_cryptlib; //Synapse
19145: 845 uPSI_ssl_openssl; //Synapse
19146: 846 uPSI_synhttp_daemon; //Synapse
19147: 847 uPSI_NetWork; //mX4
19148: 848 uPSI_PingThread; //Synapse
19149: 849 uPSI_JvThreadTimer; //JCL
19150: 850 unit uPSI_wwSystem; //InfoPower
19151: 851 unit uPSI_IdComponent; //Indy
19152: 852 unit uPSI_IdIOHandlerThrottle; //Indy
19153: 853 unit uPSI_Themes; //VCL

```

```

19154: 854 unit uPSI_StdStyleActnCtrls;           //VCL
19155: 855 unit uPSI_UDDIHelper;                   //VCL
19156: 856 unit uPSI_IdIMAP4Server;                //Indy
19157:
19158:
19159:
19160:
19161:
19162: //////////////////////////////////////
19163: //Form Template Library FTL
19164: //////////////////////////////////////
19165:
19166: 26 FTL For Form Building out of the Script, eg. 399_form_templates.txt
19167:
19168: 045 unit uPSI_VListView                      TFormListView;
19169: 263 unit uPSI_JvProfiler32;                  TProfReport
19170: 270 unit uPSI_ImgList;                       TCustomImageList
19171: 278 unit uPSI_JvImageWindow;                 TJvImageWindow
19172: 317 unit uPSI_JvParserForm;                  TJvHTMLParserForm
19173: 497 unit uPSI_DebugBox;                      TDebugBox
19174: 513 unit uPSI_ImageWin;                      TImageForm, TImageForm2
19175: 514 unit uPSI_CustomDrawTreeView;            TCustomDrawForm
19176: 515 unit uPSI_GraphWin;                     TGraphWinForm
19177: 516 unit uPSI_actionMain;                   TActionForm
19178: 518 unit uPSI_CtlPanel;                     TAppletApplication
19179: 529 unit uPSI_MDIEdit;                      TEditForm
19180: 535 unit uPSI_CoolMain; {browser}            TWebMainForm
19181: 538 unit uPSI_frmExportMain;                 TSynexportForm
19182: 585 unit uPSI_usniffer; {PortScanForm}       TSniffForm
19183: 600 unit uPSI_ThreadForm;                   TThreadSortForm;
19184: 618 unit uPSI_delphi_arduino_Unit1;         TLEDForm
19185: 620 unit uPSI_fplotMain;                    TfplotForm1
19186: 660 unit uPSI_JvDBQueryParamsForm;          TJvQueryParamsDialog
19187: 677 unit uPSI_OptionsFrm;                   TfrmOptions;
19188: 718 unit uPSI_MonForm;                      TMonitorForm
19189: 742 unit uPSI_SimplePortMain;               TPortForm1
19190: 770 unit uPSI_ovccalc;                      TOvcCalculator //widget
19191: 810 unit uPSI_DbExcept;                     TDbEngineErrorDlg
19192: 812 unit uPSI_GL_actorUnit1;                TglActorForm1 //OpenGL Robot
19193: 846 unit uPSI_synhttp_daemon;               TTCPhHttpDaemon, TTCPhHttpThrd, TPingThread
19194:
19195:
19196:
19197: ex.:with TEditForm.create(self) do begin
19198:   caption:= 'Template Form Tester';
19199:   FormStyle:= fsStayOnTop;
19200:   with editor do begin
19201:     Lines.LoadFromFile(Exepath+'\\docs\\Readme_rus_mX2.rtf
19202:     SelStart:= 0;
19203:     Modified:= False;
19204:   end;
19205: end;
19206: with TWebMainForm.create(self) do begin
19207:   URLs.Text:= 'http://www.kleiner.ch';
19208:   URLs.Click(self); Show;
19209: end;
19210: with TSynexportForm.create(self) do begin
19211:   Caption:= 'Synexport HTML RTF tester';
19212:   Show;
19213: end;
19214: with TThreadSortForm.create(self) do begin
19215:   showmodal; free;
19216: end;
19217:
19218: with TCustomDrawForm.create(self) do begin
19219:   width:=820; height:=820;
19220:   image1.height:= 600; //add properties
19221:   image1.picture.bitmap:= image2.picture.bitmap;
19222:   //SelectionBackground1Click(self) CustomDraw1Click(self);
19223:   Background1.click;
19224:   bitmap1.click;
19225:   Tile1.click;
19226:   Showmodal;
19227:   Free;
19228: end;
19229:
19230: with TfplotForm1.Create(self) do begin
19231:   BtnPlotClick(self);
19232:   Showmodal; Free;
19233: end;
19234:
19235: with TOvcCalculator.create(self) do begin
19236:   parent:= aForm;
19237:   //free;
19238:   setbounds(550,510,200,150);
19239:   displaystr:= 'maXcalc';
19240: end;
19241:
19242:

```

```

19243:
19244:
19245:
19246:
19247: //////////////////////////////////////
19248: All maXbox Tutorials Table of Content 2014
19249: //////////////////////////////////////
19250: Tutorial 00 Function-Coding (Blix the Programmer)
19251: Tutorial 01 Procedural-Coding
19252: Tutorial 02 OO-Programming
19253: Tutorial 03 Modular Coding
19254: Tutorial 04 UML Use Case Coding
19255: Tutorial 05 Internet Coding
19256: Tutorial 06 Network Coding
19257: Tutorial 07 Game Graphics Coding
19258: Tutorial 08 Operating System Coding
19259: Tutorial 09 Database Coding
19260: Tutorial 10 Statistic Coding
19261: Tutorial 11 Forms Coding
19262: Tutorial 12 SQL DB Coding
19263: Tutorial 13 Crypto Coding
19264: Tutorial 14 Parallel Coding
19265: Tutorial 15 Serial RS232 Coding
19266: Tutorial 16 Event Driven Coding
19267: Tutorial 17 Web Server Coding
19268: Tutorial 18 Arduino System Coding
19269: Tutorial 18_3 RGB LED System Coding
19270: Tutorial 19 WinCOM /Arduino Coding
19271: Tutorial 20 Regular Expressions RegEx
19272: Tutorial 21 Android Coding (coming 2013)
19273: Tutorial 22 Services Programming
19274: Tutorial 23 Real Time Systems
19275: Tutorial 24 Clean Code
19276: Tutorial 25 maXbox Configuration I+II
19277: Tutorial 26 Socket Programming with TCP
19278: Tutorial 27 XML & TreeView
19279: Tutorial 28 DLL Coding (available)
19280: Tutorial 29 UML Scripting (coming 2014)
19281: Tutorial 30 Web of Things (coming 2014)
19282: Tutorial 31 Closures (coming 2014)
19283: Tutorial 32 SQL Firebird (coming 2014)
19284:
19285:
19286: ref Docu for all Type Class and Const in maXbox_types.pdf
19287: using Docu for this file is maxbox_functions_all.pdf
19288: PEP - Pascal Education Program Lib Lab
19289:
19290:
19291: http://stackoverflow.com/tags/pascalscript/hot
19292: http://www.jrsoftware.org/ishelp/index.php?topic=scriptfunctions
19293: #locs:15162
19294: http://sourceforge.net/apps/mediawiki/maXbox
19295: http://www.blaisepascal.eu/
19296: https://github.com/maxkleiner/maXbox3.git
19297: http://www.heise.de/download/maxbox-1176464.html
19298: http://www.softpedia.com/get/Programming/Other-Programming-Files/maXbox.shtml
19299: https://www.facebook.com/pages/Programming-maXbox/166844836691703
19300:
19301: ---- bigbitbox code_cleared_checked----
```